Forêts déductibles maximales

Miklos Molnar, Raymond Marie

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Forêts déductibles maximales

Miklós Molnár et Raymond Marie

N°3974

Juillet 2000

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--3974--FR

INRIA
RENNES

Forêts déductibles maximales

Miklós Molnár* et Raymond Marie†

Thème 1 — Réseaux et systèmes
Projet ARMOR

Rapport de recherche n° 3974 — Juillet 2000 — 41 pages

Résumé : Certaines heuristiques de construction d'arbres couvrants partiels de poids minimal d'un ensemble de nœuds soulèvent la question suivante : quelle partie de l'arbre d'origine faut-il supprimer lors de l'amélioration d'un arbre couvrant partiel existant par ajout d'un sous-arbre couvrant partiel plus avantageux ? Pour les algorithmes travaillant sur un graphe complet, une réponse (facilement calculable) a été formulée.

Dans notre travail, nous généralisons le problème. Dans le cas général, la structure qui peut être supprimée d'un arbre après l'ajout d'un autre arbre (pour retrouver un arbre couvrant un ensemble donné de nœuds) est une forêt. Nous donnons ici une méthode de réduction du problème posé ainsi que la solution algorithmique pour identifier la forêt déductible maximale.

Mots-clé : Optimisation combinatoire, problème de Steiner, forêt déductible.

(Abstract: pto)

* {Miklos.Molnar}@irisa.fr
† {Raymond.Marie}@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Maximal Removal Forests

Abstract: Some heuristics for solving the partial minimum spanning tree problem raise the following question: when we are improving an existing partial spanning tree by addition of a (more advantageous) partial subtree, which part of the original tree can be removed? For the algorithms working on a complete graph, an easily computable answer has been already formulated.

In our work, we generalize the problem. In the general case, the structure which can be removed from a tree after the addition of another one (to find a spanning tree covering a given set of nodes) is a forest. Here, we give a reduction method of the arising problem as well as the algorithmic solution to identify the maximum removal forest.

Key-words: Combinatorial optimisation, Steiner problem, removal forest.

1 Introduction

Plusieurs heuristiques connues proposent de construire un arbre couvrant partiel de longueur acceptable en procédant par améliorations successives d'arbres couvrants existants, faciles à calculer.

L'amélioration consiste à prendre un sous-ensemble de nœuds, à trouver l'arbre minimal de Steiner et à remplacer la partie déterminée par le dit sous-ensemble par l'arbre minimal. Ce qui fournit un résultat de longueur inférieure.

Le lecteur peut trouver des exemples de ce genre d'heuristique parmi les travaux de A. Zelikovsky (cf. [1] et [3] pour des exemples) ou de P. Berman et de V. Ramaiyer (cf. dans [4]).

La démarche commune à ces heuristiques peut se résumer ainsi :

- Il existe déjà un arbre couvrant partiel pour un groupe R de nœuds (souvent, on suppose que le graphe dans lequel on cherche l'arbre couvrant partiel est un graphe complet),
- pour un sous-ensemble de nœuds de cet arbre, on propose un arbre couvrant différent (en général, on cherche un arbre plus favorable en longueur),
- l'ajout du nouvel arbre introduit des circuits dans l'arbre couvrant partiel de R,
- pour rétablir un arbre couvrant partiel de R, la suppression de certaines arêtes est nécessaire,
- afin d'avoir un gain maximal pour cette opération, il faut supprimer l'ensemble d'arêtes déductibles dont la somme des longueurs est maximale.

Lors de précédentes études, nous avons examiné des heuristiques de construction d'arbres couvrant partiels à l'aide d'unions successives de sous-arbres déjà connus (cf. [5] et [6]). Les algorithmes proposés dans ces études connectent des sous-arbres à l'aide d'arbres minimaux de Steiner dont les feuilles sont partagées entre les sous-arbres à connecter ; un plus court chemin peut être considéré comme la connexion la plus simple : cet arbre minimal de Steiner n'a que deux feuilles dans les sous-arbres à relier. Si la connexion possède plus de deux feuilles dans un sous-arbre isolé au départ, des circuits se produisent lors de l'opération de la connexion.

Dans la suite, nous supposons que la longueur d'un arbre, d'un ensemble d'arêtes, ou plus généralement la longueur $d(G)$ d'un graphe G valué est la somme des valeurs de ses arêtes. Dans les deux cas mentionnés ci-dessus, le but des heuristiques est la diminution de la longueur $d(T)$ d'un arbre couvrant T d'un groupe R de nœuds. Pour cela, on ajoute une structure supplémentaire à l'arbre et on supprime un ensemble d'arêtes. Dans la suite, nous allons voir que l'ensemble des arêtes pouvant être supprimé correspond à une forêt. On cherche alors la forêt de longueur maximale qui peut être enlevée d'un arbre couvrant partiel dans le contexte mentionné. Dans la section 2 nous traitons le cas des heuristiques travaillant sur les graphes complets et nous présentons le problème général ainsi que son analyse dans la section 3. L'algorithme proposé sur la base de notre analyse est résumé dans la section 4 qui contient également la détermination de la complexité de celui-là.

2 Suppression des circuits pour un graphe complet

Les graphes *complets* considérés dans cette partie correspondent à la *fermeture métrique* de graphes connexes valués par des valeurs positives. Il est important de noter que, pour la classe de graphes complets et métriques, il n'existe pas de chemin de longueur supérieure où égale à deux de valeur plus courte que la valeur d'une arête entre deux nœuds (inégalité triangulaire).

Le problème a été étudié par Zelikovsky qui a proposé une heuristique (cf. [1]) pour choisir sur un graphe *complet* un arbre couvrant partiel d'un groupe de nœuds R. Sa technique consiste à envisager l'utilisation d'arbres minimaux de Steiner de triplets appartenant au groupe R. Le résultat de l'ajout d'un arbre couvrant d'un triplet à un autre arbre est un graphe qui contient deux circuits indépendants. Pour retrouver un arbre couvrant partiel du groupe de nœuds R (par exemple un arbre de longueur minimale), la suppression de certaines arêtes est nécessaire. L'algorithme de Zelikovsky a été généralisé (cf. [2], [3]) en envisageant l'utilisation d'arbres minimaux de Steiner de n-uplets de nœuds pris dans R. Il est connu que l'ajout d'un arbre couvrant d'un n-uplet produit un graphe contenant $n-1$ circuits indépendants. Pour retrouver un arbre couvrant partiel du groupe de nœuds R, il faut supprimer des arêtes et si possible celles qui rélisent une somme maximale de longueurs. Comme le graphe est complet et métrique, il est facile de prouver que l'élimination des $n-1$ circuits nécessite la suppression de $n-1$ arêtes de l'arbre couvrant initial.

Pour améliorer les arbres approchés d'un arbre minimal de Steiner dans un graphe complet, Berman et Ramaiyer ont proposé un autre algorithme [4]. Leur algorithme utilise un marquage des arêtes déductibles qui est basé sur le lemme présenté au paragraphe suivant.

2.1 Formulation du problème

Soit $G = (V,E,d)$ un graphe complet non orienté et valué par des valeurs positives, $R \subset V$ un sous-ensemble de nœuds et $T = (R,E_T)$ un arbre couvrant de R dans G. Soit $\tau \subset R$ un sous-ensemble de R. L'ajout à T d'un arbre minimal de Steiner C de τ induit des circuits. Pour espérer retrouver un arbre couvrant R de longueur inférieure à $d(T)$, un ensemble d'arêtes de longueur maximale doit être supprimé de T.

Pour y arriver, nous allons utiliser le lemme suivant dû à Berman et Ramaiyer.

Lemme 2.1 *Soit e une arête de coût maximal telle que chaque composante connexe de $T - \{e\}$ contienne au moins un nœud de τ. En ajoutant C à T, il existe un ensemble d'arêtes de coût maximal contenant e pouvant être supprimé de T et qui rétablit un arbre couvrant de R dans G.*

Le lecteur pourra trouver la preuve de ce lemme dans [4].

2.2 Algorithme de suppression proposé

Le lemme 2.1 traite le cas où l'arbre couvrant d'un sous-ensemble τ est ajouté à l'arbre $T = (R,E_T)$ couvrant le sous-ensemble R de nœuds d'un graphe complet. Dans ce cas,

($|\tau| - 1$) arêtes de E_T peuvent être sélectionnées et supprimées de T pour obtenir un nouvel arbre couvrant. On peut sélectionner l'ensemble d'arêtes de coût maximal à supprimer de la façon suivante.

Lemme 2.2 *Le lemme 2.1 peut être appliqué itérativement pour déterminer l'ensemble des arêtes à supprimer par des sélections successives de l'arête de longueur maximale.*

Preuve. Soit e l'arête qui correspond aux critères du lemme 2.1. Sa suppression élimine un circuit et éclate T en deux sous-arbres notés T_1 et T_2. Chacun des deux contient au moins un nœud de τ. Si T_i ne possède qu'un seul nœud de τ, alors il n'est plus la cause d'aucun circuit. S'il contient plusieurs nœuds de τ, alors pour éliminer les circuits engendrés, on peut appliquer le lemme 2.1 de nouveau, puisque T_i est l'arbre couvrant de ses nœuds et uniquement de ses nœuds R_i. On obtient un problème réduit du même genre. ■

Supposons que E' soit la liste des arêtes de T triée selon l'ordre décroissant des longueurs. La sélection des arêtes peut s'effectuer de la façon suivante :

Suppression des circuits (dans un graphe complet, métrique)

```
 e = entête(E') ;
 k = 1 ;
 E = E_T ;
Jusqu'à k = |τ| − 1 faire :
      Si chaque composante de T' = (R,E\{e}) contient un nœud de τ
           E = E\{e} ; /* suppression de e */
           T = (R,E) ;
           k = k + 1 ;
      e = suivant(E') ; /* arête suivante */
fait ;
fin.
```

Remarque : Comme les graphes complets considérés dans ce paragraphe correspondent à la fermeture métrique de graphes connexes quelconques, un arbre de Steiner d'un sous-ensemble R de nœuds ne contient que des nœuds de R. Dans le cas de l'ajout d'un nouvel arbre sur un arbre couvrant partiel d'un graphe quelconque, l'algorithme décrit ci-dessus ne garantit pas la suppression de la partie la plus globalement favorable de la structure commune. Nous illustrons cette impasse de l'algorithme à l'aide du contre-exemple de la figure 1. Sur cette figure, l'arbre C dessiné en pointillé est ajouté à l'arbre T. Si l'on applique le lemme de Berman et Ramaiyer, et que l'on supprime les arêtes selon l'algorithme ci-dessus, l'arête de longueur 8 et une autre de longueur 4 seront supprimées. En fait, on constate intuitivement que la solution optimale del'élimination des circuits à l'aide d'un ensemble d'arêtes de longueur maximale correspond à l'élimination des arêtes indiquées en gras sur la figure ; ces dernières représentant une longueur déductible égale à 13.

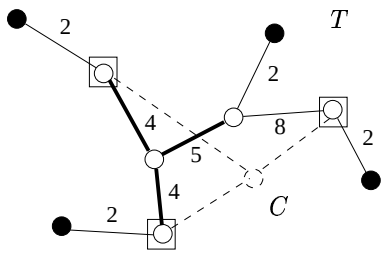


FIG. 1 – *Suppression des circuits dans le cas d'un arbre couvrant quelconque*

3 Suppression des circuits dans des arbres de Steiner quelconques

3.1 Définitions nécessaires et formulation du problème

Dans le cas général, l'arbre T contient des nœuds qui n'appartiennent pas à l'ensemble R à couvrir. Il s'agit alors de supprimer des chemins de T tels que la somme des coûts soit la plus élevée possible, tout en gardant un arbre couvrant des nœuds de R. La partie qui peut être supprimée de T correspond maintenant à un ensemble de sous-arbres, i.e., à une forêt. Notons que par hypothèse, la suppression des circuits doit se faire sur l'arbre couvrant T de l'ensemble R (et non sur l'arbre C).

Définition 3.1 (Soustraction minimale (d'un sous-arbre relativement à un arbre)) *Soit $T = (V_T,E_T)$ un arbre et $A = (V_A,E_A)$ un de ses sous-arbres; c-à-d : $V_A \subseteq V_T$, $E_A \subseteq E_T$. Indiquons l'ensemble des nœuds internes de A qui possèdent le même degré dans A et dans T par V_{A_s}. Définissons l'opération $T \ominus A$ de la façon suivante :*

$$T \ominus A = F = (V_F,E_F)$$

où $V_F = V_T \backslash V_{A_s}$ et $E_F = E_T \backslash E_A$.

La figure 2 illustre cette opération. Notons que, si $F_1 = \{T_i,\ i = 1,...,|F_1|\}$ est une forêt (les arbres d'une forêt sont disjoints), la définition se généralise dans le cas où $\exists\ T_j \in F_1$ tel que $A \subseteq T_j$. Dans ce cas l'opération $F_1 \ominus A$ donne la forêt $F_2 = \{F_1 \backslash T_j\} \cup \{T_j \ominus A\}$.

Propriétés : L'opérateur $\ominus$ possède les propriétés suivantes :

P1 : Le résultat F est toujours une forêt.

P2 : L'opération $T \ominus T$ donne une forêt qui contient les feuilles isolées de T (et non l'élément nulle).

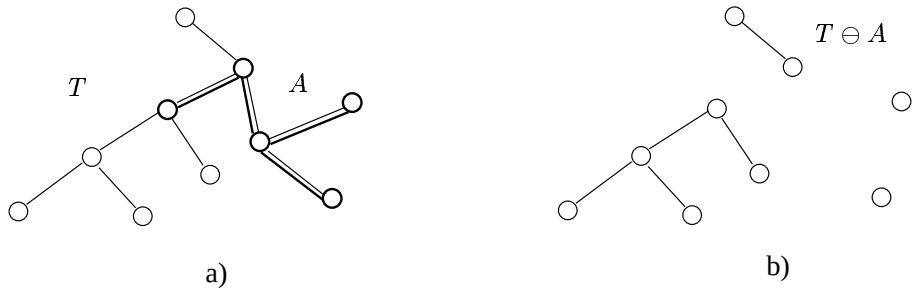


FIG. 2 – *Soustraction d'un sous-arbre d'un arbre*

P3 : La longueur du résultat est $d(F) = d(T) - d(A)$.

P4 : Soit A_1 et A_2 deux sous-arbres disjoints de T. L'operateur $\ominus$ est commutatif dans le sens où $T \ominus A_1 \ominus A_2 = T \ominus A_2 \ominus A_1$.

Preuve. Les preuves de P2 et P3 sont triviales car elles découlent directement de la définition. Pour prouver P1, il suffit de rappeler qu'un arbre est un cas particulier d'une forêt et qu'un arbre privé d'une ou de plusieurs de ses arêtes est une forêt.

Pour prouver P4, remarquons d'abord que, les sous-arbres étant disjoints, A_1 est inclus dans un arbre unique de $T \ominus A_2$ et A_2 est inclus dans un arbre unique de $T \ominus A_1$. Ensuite, notons $F_i = T \ominus A_i$ et désignons par T_{ij} l'arbre de F_i contenant A_j.

Alors on peut écrire $F_i \ominus A_j = \{ \{F_i \backslash T_{ij}\} \cup T_{ij} \ominus A_j \} = \{ \{F_i \backslash T_{ij}\} \cup \{F_j \backslash T_{ji}\} \}$.

De le même manière, $F_j \ominus A_i = \{ \{F_j \backslash T_{ji}\} \cup T_{ji} \ominus A_i \} = \{ \{F_j \backslash T_{ji}\} \cup \{F_i \backslash T_{ij}\} \}$; les deux forêts sont donc identiques.

■

Remarque : On pourrait aussi définir une soustraction maximale (plus habituelle), en prenant $V_F = V_T \backslash \{V_{A_s} \cup V_{A_f}\}$, où V_{A_f} désigne l'ensemble des feuilles de A qui sont aussi feuilles de T.

L'application en cascade de l'opérateur $\ominus$ produit toujours une forêt (cela découle par induction de la propriété P1). Toutefois, pour que la cascade d'opérations $A_1 \ominus A_2 \ominus \cdots \ominus A_{i-1} \ominus A_i$ ait du sens, il faut que l'arbre A_i soit inclus dans l'un des arbres de la forêt obtenue par la cascade d'opérations $A_1 \ominus A_2 \ominus \cdots \ominus A_{i-1}$. La cascade d'opérations $T \ominus T \ominus \cdots \ominus T \ominus T$ n'a de sens que si T est constitué d'un seul nœud.

Définition 3.2 (Soustraction d'une forêt à un arbre) *Soit $T = (V_T, E_T)$ un arbre et $F = \{A_i, i = 1,...,|F|\}$ une forêt composée de sous-arbres de T. Définissons l'opération $T \ominus F$ à l'aide des soustractions successives :*

$$T \ominus F = F' = T \ominus A_1 \ominus A_2 \ominus \cdots \ominus A_{|F|}$$

Remarques : Selon la propriété P4 de la soustraction minimale, le résultat ne dépend pas de l'ordre d'énumération des arbres de la forêt F. On pourrait également élargir la définition de la soustraction et définir le résultat de la différence de deux forêts (mais cette généralisation ne sera pas utilisée ici).

Lemme 3.3 *Un arbre privé de son sous-arbre $A = (V_A,E_A)$ (dans le sens de l'opérateur $\ominus$ défini ci-dessus) est une forêt qui contient $|V_A| - |V_{A_s}|$ arbres isolés.*

Preuve. Si l'on soustrait un sous-arbre A d'un arbre T, le résultat est une forêt. D'une part - selon la définition 3.1 de la soustraction - seuls les nœuds internes qui possèdent le même degré dans T et dans A sont enlevés de T. Les autres nœuds de A appartiennent à un arbre du résultat chacun. Il y a $|V_A| - |V_{A_s}|$ nœuds de A qui sont préservés par la soustraction. D'autre part, la soustraction élimine toutes les arêtes de A; il ne reste pas de connexion entre les nœuds de A sauvegardés par l'opération $T \ominus A$. Il existe donc autant d'arbres dans le résultat que de nœuds préservés de A. ■

Définition 3.4 (Arbre déductible - arbre déductible maximal) *Soit $T = (V_T,E_T)$ un arbre couvrant un groupe R sur le graphe G et $C = (V_C,E_C)$ un autre arbre sur G n'ayant pas d'arête commune, mais des feuilles communes avec T ; c-à-d : $\tau = V_T \cap V_C$, $\tau \neq \emptyset$ et $E_T \cap E_C = \emptyset$. Un sous-arbre $T_d = (V_{T_d},E_{T_d})$ est déductible de T relativement à τ, si $V_{T_d} \cap R = \emptyset$ et si chacun des composants isolés de la forêt $T \ominus T_d$ contient au moins un nœud de τ.*

L'arbre $T_d^\star$ déductible maximal de T relativement à τ est l'arbre de longueur maximale parmi les arbres déductibles de T :

$$d(T_d^\star) = \max_{T_d \in \mathcal{T}_d} d(T_d)$$

où $\mathcal{T}_d$ indique l'ensemble des arbres déductibles de T relativement à τ.

Définition 3.5 (Forêt déductible - forêt déductible maximale) *Soit T et C deux arbres et τ un ensemble de nœuds définis sur G comme ci-dessus. Supposons qu'une forêt $F_d = \{A_i, i = 1,...,f_d\}$ se compose des sous-arbres A_i disjoints de T, c-à-d :*

$$\begin{array}{rcll} A_i & \subset & T & \forall i \\ F_d & = & \bigcup_{i=1}^{f_d} A_i & \\ A_i \bigcap A_j & = & \emptyset & \forall i,j,\ i \neq j \end{array}$$

La forêt F_d est déductible de T relativement à τ, si la forêt $F_j = T \ominus F_d$ contient tous les nœuds de R et chacun des composants isolés de F_j contient au moins un nœud de τ.

La forêt $F_d^\star$ déductible maximale de T relativement à τ est la forêt de longueur maximale parmi les forêts déductibles de T :

$$d(F_d^\star) = \max_{F_d \in \mathcal{F}_d} d(T_d)$$

Ici $\mathcal{F}_d$ est l'ensemble des forêts qui peuvent être déduites de T relativement à τ.

Propriété : Une forêt déductible ne contient que des arbres déductibles.

Preuve. Supposons qu'une forêt F déductible de T relativement à τ contienne un arbre T_k non déductible relativement à τ. Si l'on déduit cet arbre en premier de T, on obtient une forêt dont un des critères de la définition 3.4 n'est pas vérifié : la suppression ne préserve pas tous les nœuds de R ou chaque arbre isolé ne contient pas un élément de τ. Quand on enlève les autres arbres de la forêt, le critère de la définition 3.4 non satisfait par T_k reste non vérifié; ainsi, la forêt qui contient T_k ne peut pas être une forêt déductible. ■

L'inverse de cette propriété n'est pas vrai. Le fait qu'une forêt se compose uniquement d'arbres déductibles d'un arbre T relativement à un ensemble de nœuds τ ne signifie pas que la forêt est une forêt déductible de T relativement à τ. Le lecteur peut trouver facilement des contre-exemples.

En résumé, le problème étudié ici se formule de la façon suivante :
On considère un arbre $T = (V,E)$ valué positivement qui couvre un ensemble $R \subset V$ de nœuds et on considère l'ensemble τ des feuilles de l'arbre $C = (V_C,E_C)$ qui touche l'arbre T. Par hypothèse, les feuilles de T appartiennent à R (T ne contient pas de branche inutile). Nous avons vu que la juxtaposition de T et de C introduit des circuits dans la structure couvrante de R. Dans le but d'obtenir à nouveau un arbre couvrant de R, une forêt déductible maximale de T relativement à τ doit être trouvée, afin d'éliminer les circuits générés par l'adjonction de C.

Le calcul de la forêt déductible maximale est possible. Toutefois, on gagnera en efficacité si plusieurs simplifications éventuelles sont effectuées.De telles simplifications sont présentées au paragraphe suivant, après avoir introduit de nouvelles notions.

3.2 Analyse et simplifications possibles

Lemme 3.6 *Si l'on enlève de l'arbre T une forêt F_d déductible maximale relativement à τ, on obtient une forêt $F_j = T \ominus F_d = \{A_i, i = 1,...,|\tau|\}$, qui est une forêt joignante minimale (cf. [6]). Chaque arbre A_i de F_j contient exactement un élément de τ et un arbre A_i possèdant plus d'un nœud possède aussi au moins un nœud de R.*

Preuve.

i) Puisque la suppression de la forêt F_d est réalisée à l'aide de l'opérateur $\ominus$, le résultat est une forêt.

ii) Chaque arbre du résultat doit contenir une et une seule des feuilles appartenant à τ. En effet, supposons qu'il existe un arbre dans le résultat qui ne possède pas d'élément de τ. Cela contredit le fait que la forêt F_d supprimée est une forêt déductible relativement à τ. Supposons maintenant qu'il existe un arbre $A_i^{'}$ qui contienne plusieurs éléments de τ. Dans ce cas, cet arbre contient aussi au moins un sous-arbre T_{A_i} déductible relativement à τ qui coupe $A_i^{'}$ en plusieurs composantes connexes. Cet arbre T_{A_i} peut être ajouté à la forêt déductible F_d et $F_d \cup T_{A_i}$ reste une forêt déductible relativement à τ ; ce qui contredit le fait que F_d est une forêt déductible maximale.

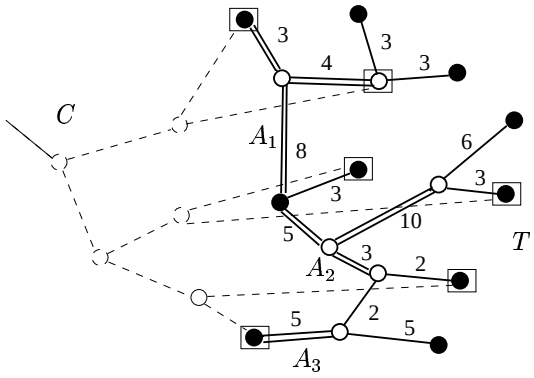


FIG. 3 – *Forêt de coût maximum à supprimer*

iii) Chaque arbre du résultat qui possède au moins deux nœuds contient au moins un nœud de R. Selon la définition de la soustraction, F_j est une forêt qui couvre tous les nœuds de R. Admettons qu'il existe un arbre A_k dans $F_j = T \ominus F_d$ qui contienne une arête mais qui ne contienne pas de nœud de R. Cet arbre peut être rajouté à F_d. De cette façon, $T \ominus F_d \ominus A_k$ couvre R et $F_d \cup A_k$ est une forêt déductible, ce qui contredit le fait que F_d est une forêt déductible maximale. ■

La figure 3 montre un arbre couvrant T d'un ensemble de nœuds de R qui est connecté par un arbre C. La partie qui est susceptible d'être supprimée est une partie de T limitée par les feuilles de C. Sur la figure, les arbres A_i, i = 1,2,3 de la forêt déductible maximale sont indiqués (les arêtes de ces arbres sont représentées par deux lignes). Afin de rétablir un arbre couvrant de longueur minimale après la connexion avec C, il faut éliminer la forêt déductible maximale. Nous allons maintenant déterminer une forêt déductible maximale de T relative à l'ensemble τ de ses nœuds à l'aide d'une technique de décomposition (cf. paragraphe 3.2.1). Ce n'est qu'au paragraphe 3.2.2 que nous présentons les simplifications (potentielles) qu'il faudra également effectuer pour gagner en efficacité.

3.2.1 Partitionnement de l'arbre couvrant

Définition 3.7 (Quasi-partition d'un arbre engendrée par un ensemble de nœuds) *Soit $T = (V_T, E_T)$ un arbre et $\tau \subset V_T$ un sous-ensemble de ses nœuds. Soit un nœud $v \in \tau$ considéré comme la racine de T. Si v est de degré δ, considérons les δ sous-arbres issus de v en dupliquant le nœud v. Ces sous-arbres sont non disjoints car ils partagent le nœud v. On parle alors de la quasi-partition de T engendrée par v. La quasi-partition de T engendrée par τ est l'ensemble des sous-arbres de T obtenus par répétition de l'opération relativement à tous les nœuds appartenant à τ.*

La figure 4 montre un exemple de la quasi-partition d'un arbre couvrant T. Les nœuds à couvrir sont indiqués par des cercles pleins et les nœuds de l'ensemble τ par des carrés.

Lemme 3.8 *Soit P la quasi-partition de T engendrée par τ. Indiquons par η_δ le nombre de nœuds de τ de degré δ dans l'arbre T. Soit δ_{max} le degré maximal parmi les degrés des nœuds appartenant à τ. La cardinalité de la quasi-partition P est donnée par :*

$$|P| = 1 + \sum_{\delta=2}^{\delta_{max}} (\delta - 1)\eta_\delta$$

Si l'ensemble τ contient plusieurs éléments ($|\tau| > 1$), la quasi-partition engendrée par τ contient au moins un sous-arbre qui possède au moins deux nœuds de τ.

Preuve. Considérons la construction de la quasi-partition P par répétition de l'opération relativement à tous les nœuds appartenant à τ . Etant donné que T est un arbre, la première feuille τ_1 le coupe en *degré*(τ_1) sous-arbres. Pour la i-ème opération : soit δ le degré du nœud τ_i. L'opération sur ce nœud ne concerne qu'un seul sous-arbre T_j de la quasi-partition déjà existante et le décompose en δ sous-arbres ; c-à-d : il augmente le nombre de sous-arbres dans la quasi-partition de $(\delta - 1)$. (Si τ_i est une feuille dans T_j (i.e., $\delta = 1$), alors le nombre de sous-arbres dans la quasi-partition ne change pas.) Finalement, le nombre de sous-arbres dans la quasi-partition est égale à :

$$|P| = 1 + \sum_{\delta=2}^{\delta_{max}} (\delta - 1)\eta_\delta$$

Quand on traite le deuxième élément τ_2 de τ, on le trouve sur un sous-arbre de la quasi-partition qui contient déjà un élément τ_1 de τ. Un des sous-arbres issu de l'opération par τ_2 contient aussi τ_1. Cette nouvelle division produit alors un composant qui possède au moins deux nœuds de τ. ■

Lemme 3.9 *Les sous-arbres d'une quasi-partition P engendrée par τ n'ayant qu'une seule feuille appartenant à τ ne contiennent pas de forêt déductible.*

Chaque sous-arbre de la quasi-partition qui possède au moins deux nœuds de τ contient des forêts déductibles dont au moins une forêt déductible maximale.

Preuve. Les feuilles des sous-arbres d'une quasi-partition sont des éléments de l'ensemble R à couvrir ou des éléments de τ (qui ne peuvent être que des feuilles dans les composants de la quasi-partition). Admettons qu'on supprime une arête d'un arbre qui ne contient qu'une feuille appartenant à τ. Cette suppression coupe l'arbre en deux : une des deux parties contient la feuille de τ. L'autre partie contient au moins une feuille du composant avant la suppression qui est obligatoirement un nœud de R. Ce composant et ce nœud restent isolés après la suppression de l'arête. Ainsi, l'arête (et de cette façon toute arête du composant) ne peut être supprimée.

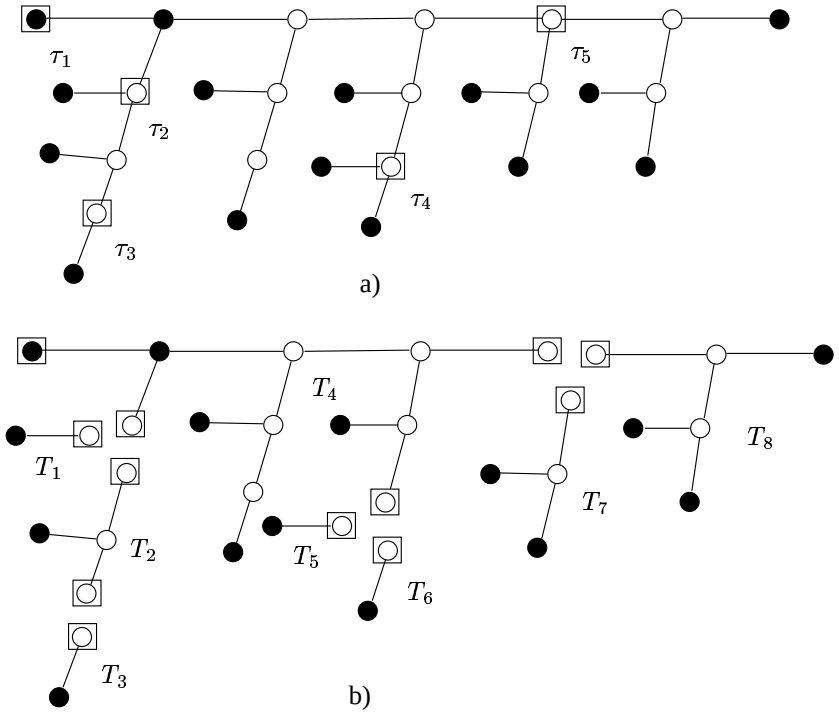


FIG. 4 – *Quasi-partition d'un arbre couvrant*

Supposons qu'il n'y ait pas de forêt déductible dans un composant T_j qui possède deux nœuds $\tau_1 \in \tau$ et $\tau_2 \in \tau$. τ_1 et τ_2 sont connectés dans T_j mais aussi dans C qui implique une circuit. Il existe au moins un sous-arbre (un chemin) dans T_j qui peut être supprimé. Si k feuilles de τ se trouvent dans T_j, le nombre des circuits indépendants est égale à $k-1$ et des forêts déductibles existent. Sur l'ensemble fini des forêts, il existe toujours au moins une forêt de longueur maximale. ■

Lemme 3.10 *Dans une quasi-partition engendrée par un ensemble τ, le nombre de sous-arbres contenant au moins deux éléments de τ est inférieur à $|\tau|$.*

Preuve. Considérons le sous-ensemble $\mathcal{T}$ des sous-arbres de la quasi-partition qui contiennent au moins deux éléments de τ chacun. La réunification de ces sous-arbres donne un arbre, un sous-arbre de l'arbre d'origine.

L'union de deux sous-arbres de $\mathcal{T}$ se réalise par superposition des nœuds de τ communs aux deux sous-arbres. En commençant par un arbre arbitraire de $\mathcal{T}$, ajoutons successivement les autres arbres à l'union déjà faite. Si au moment d'une réunification il existait deux nœuds communs entre les deux arbres, l'union produirait un cycle. Donc, à chaque pas de

la réunification, les deux arbres unifiés ne peuvent avoir qu'un seul nœud commun. Puisque chaque sous-arbre contient au moins deux nœuds de τ, on ne peut avoir plus de $|\tau - 1|$ sous-arbres dans le sous-ensemble $\mathcal{T}$. ■

Le lemme suivant permet d'obtenir la forêt déductible maximale de T relativement à τ par décomposition.

Lemme 3.11 *Soit $P = \{T_j,\ j = 1,...,|P|\}$ la quasi-partition de T engendrée par τ. Indiquons par $\mathcal{T}_{2+}$ l'ensemble des arbres de P qui contiennent au moins deux éléments de τ chacun. Pour chaque arbre $T_j \in \mathcal{T}_{2+}$, $j = 1,...|\mathcal{T}_{2+}|\}$, soit F_j sa forêt déductible maximale. La forêt déductible maximale F de T est l'union de ces forêts :*

$$F = \bigcup_{T_j \in \mathcal{T}_{2+}} F_j$$

Preuve.

1) Montrons qu'une forêt déductible maximale F_j de T_j appartient à une forêt déductible maximale F de T.
   Admettons qu'il n'existe pas une telle forêt dans T. Prenons alors une forêt déductible maximale F' de T. Cette dernière a un composant F'_j dans T_j qui est aussi maximal (n'oublions pas qu'on peut avoir plusieurs forêts déductibles maximales dans un arbre). Si l'on remplace F'_j par F_j, la forêt $\{F' \ominus F'_j\} \bigcup F_j$ est aussi l'une forêt déductible maximale de T, ce qui entraîne qu'une telle forêt doit exister.
2) Montrons que si l'on choisit une forêt déductible maximale F_j dans chaque composant T_j de la quasi-partition qui possède au moins deux nœuds de τ, alors l'union de ces forêts F_j donne une forêt déductible maximale F de T.
   Supposons que F ne soit pas une forêt déductible maximale ; c-à-d qu'il existe une forêt F' telle que $d(F') > d(F)$. Cette hypothèse est vérifiée, ssi il existe au moins un composant T_j de la quasi-partiton dans lequel la forêt F'_j choisie afin de construire F' est plus longue que la forêt F_j : $d(F'_j) > d(F_j)$ ce qui contredit le fait que F_j est maximale. ■

3.2.2 Réductions des composants de la quasi-partition

Selon les lemmes précédants, la forêt déductible maximale d'un arbre couvrant relative à un ensemble de ses nœuds peut être décomposée. Les composants de la quasi-partition sont des arbres dans lesquelles les éléments de τ sont des feuilles. D'après le lemme 3.9, il suffit de chercher des forêts déductibles maximales dans les arbres possédants au mois deux feuilles de τ, et le lemme 3.11 indique que l'union de ces forêts donne une forêt déductible maximale du problème de base.

Focalisons maintenant notre étude sur le cas des arbres ayant au moins deux feuilles appartenant à τ. Le calcul de la forêt déductible maximale de tels arbres restant coûteux, de possibles simplifications sont à nouveau proposées.

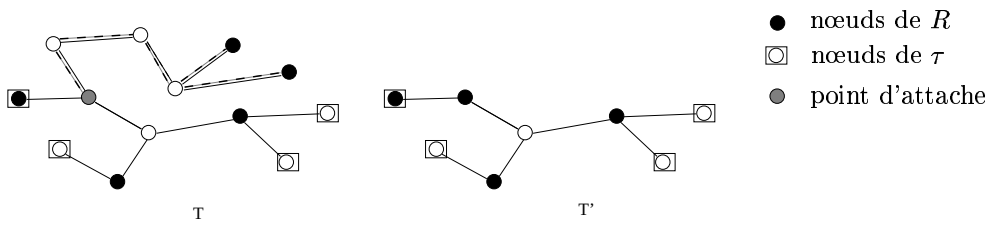


FIG. 5 – *Sous-arbre terminal pouvant être réduit*

Dans certains cas, une première réduction de l'arbre est possible ; pour celle-ci analysons les sous-arbres terminaux de l'arbre.

Définition 3.12 (Sous-arbre terminal) *Définissons par sous-arbre terminal d'un arbre T un sous-arbre T_t qui contient un et un seul nœud ayant des arêtes adjacentes hors de T_t. Un tel nœud unique est alors appelé le point d'attache du sous-arbre.*

Lemme 3.13 *Soit T un arbre d'une quasi-partition existante et appartenant à $\mathcal{T}_{2+}$. Notons R_j les éléments de R contenus dans T. Les sous-arbres terminaux de T qui n'ont pas de feuille dans τ ne sont pas déductibles.*

Preuve. Hormis le point d'attache, les feuilles d'un sous-arbre terminal sont aussi des feuilles dans T et appartiennent obligatoirement à R_j. L'arbre de la solution finale couvre également R_j et chacune des feuilles doit rester connectée à un élément de τ. Comme chaque connexion doit passer par le point d'attache du sous-arbre terminal, aucune branche du sous-arbre terminal ne peut être supprimée. ■

Grâce au lemme 3.13, le calcul de la forêt déductible maximale d'un arbre $T \in \mathcal{T}_{2+}$ de la quasi-partition devient plus simple.

Définition 3.14 (Réduction de niveau 1) *On réduit T en remplaçant les sous-arbres terminaux de longueur maximale qui n'ont pas de nœud dans τ par un nœud composé qui correspond à leur point d'attache. Comme le sous-arbre (y compris son point d'attache) est non déductible, il faut, dans la version réduite équivalente, ajouter le point d'attache à l'ensemble de nœuds R_j à couvrir. On note T' l'arbre transformé et R'_j le nouvel ensemble à couvrir.*

Remarque : Il découle de la définition ci-dessus que toutes les feuilles de l'arbre réduit T' appartiennent à τ.

La figure 5 illustre l'opération de réduction de niveau 1. Un sous-arbre terminal de longueur maximale qui ne possède pas de feuille dans τ est dessiné en pointillé sur la figure. Ce sous-arbre ne peut pas être supprimé et, pour faciliter la recherche, il peut être remplacé par sa feuille qui est le point d'attache de cet arbre vers les autres parties qui contiennent

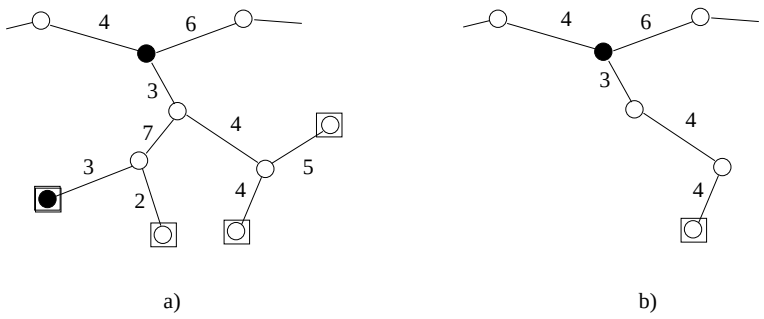


FIG. 6 – *Sous-arbre terminal réduit en une branche*

des éléments de τ. Si l'on réduit le sous-arbre terminal de longueur maximale en un nœud composé à la place de son point d'attache et si l'on ajoute ce point à l'ensemble des nœuds R_j à couvrir, on obtient un problème équivalent. Le nœud composé correspond à la notion de S-nœud (nœud de branchement) critique dans [6].

Lemme 3.15 *Soit T' un arbre couvrant de R'_j transformé d'un arbre $T \in \mathcal{T}_{2+}$ selon l'opération de réduction de niveau 1. Soit T_t un sous-arbre terminal dans T' qui ne contient pas de nœud interne appartenant à R'_j. Notons v son point d'attache. Grâce à la réduction précédente, toutes les feuilles de T_t sont dans τ. Soit b_i la branche de T_t qui relie la feuille τ_i par v et $B = \{b_i, i = 1,...,|B|\}$ l'ensemble des telles branches. Soit $b_i^\star$ la branche la moins coûteuse de B :*

$$b_i^\star = arg \min_{b_i \in B} d(b_i)$$

La partie $B \backslash \{b_i^\star\}$ appartient à la forêt déductible maximale.

Preuve. Selon la définition de la forêt déductible maximale, la soustraction des composants de la forêt préserve les nœuds de R_j et assure la présence d'un nœud de τ dans chaque arbre qui reste après la soustraction. Pour préserver v et assurer la présence d'un nœud de τ dans T_t, il suffit de garder une seule branche qui relie v à une feuille de τ. On élimine les branches les plus coûteuses de T_t en les intégrant à la forêt déductible maximale, sachant qu'on connecte v à un élément de τ par la connexion la moins coûteuse. ■

Pour simplifier le calcul de la forêt déductible maximale de T', certaines de ses branches peuvent être élaguées, s'il existe des sous-arbres vérifiant les hypothèses du lemme 3.15.

Définition 3.16 (Réduction de niveau 2) *On réduit T' en remplaçant les sous-arbres terminaux T_t de longueur maximale qui n'ont pas de nœud interne dans R'_j par leur branche la plus courte. Les autres branches élaguées de ces sous-arbres doivent être retenues pour une reconstruction ultérieure de la forêt déductible maximale. On note l'arbre transformé par T''.*

La figure 6 illustre cette opération.

Définition 3.17 (Nœuds significatifs, nœuds banals) *Dans l'arbre T'' issu de la réduction de niveau 2, soit $\mathcal{S}$ l'ensemble des S-nœuds. Qualifions de significatifs les nœuds de cet arbre qui appartiennent à τ et/ou à R'_j et/ou à $\mathcal{S}$.*

Qualifions de banals les autres nœuds possèdant un degré 2 le long des chemins qui relient les nœuds significatifs.

Définition 3.18 (Réduction de niveau 3) *Définissons l'opération de réduction dans l'arbre T'' de la façon suivante: Remplaçons les chemins de T'' qui ne contiennent pas de nœud significatif par une seule arête valuée par la longueur du chemin éliminé.*

Définition 3.19 (arbre filtré) *On appelle* arbre filtré *un arbre pris dans l'ensemble P et transformé par les réductions de niveau 1, 2 et 3 (lorsqu'elles sont possibles).*

Remarque : En tenant compte des simplifications possibles, c-à-d après les réductions de niveau 1, 2 et 3, il suffit d'analyser le cas des arbres filtrés. Dans un *arbre filtré*, toutes les feuilles sont des éléments de τ et ll n'existe que des nœuds significatifs.

La simplification d'un arbre couvrant est illustrée par la figure 7, la partie a montrant l'arbre avant les simplifications. La figure $7/b$ représente l'arbre équivalent après la réduction de niveau 1 et la figure $7/c$ présente celui obtenu après la réduction de niveau 2. Pour faciliter la compréhension, l'arbre T ne contient que les nœuds significatifs.

3.3 Forêt joignante partielle minimale

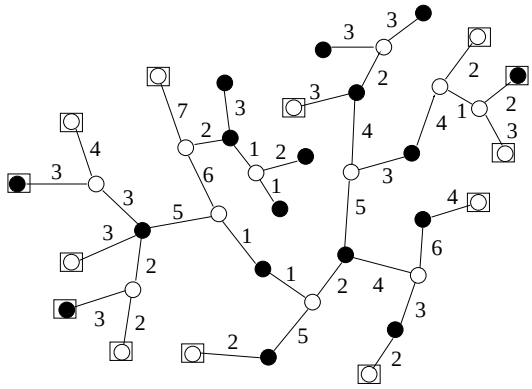
Dans cette section, nous allons baser notre démarche sur le fait que la recherche de la forêt déductible maximale et celle de la forêt joignante minimale présentée dans [6] sont deux problèmes complémentaires (i.e., pour déterminer la forêt déductible maximale, il suffit de calculer la forêt joignante minimale correspondante). Soit F_d la forêt déductible maximale de T relativement à τ et F_j la forêt joignante minimale du même arbre par rapport au même groupe de nœuds. Selon les définitions de ces forêts, elles sont complémentaires dans le sens où :

$$F_d = T \ominus F_j \tag{1}$$

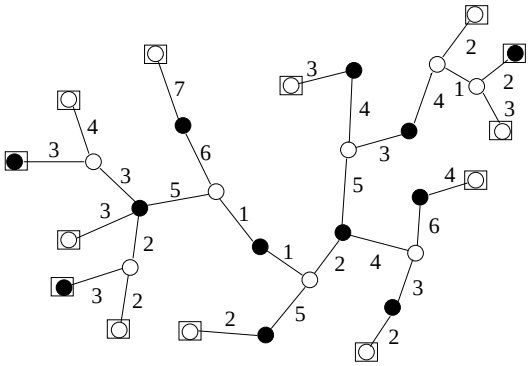
Pour calculer la forêt joignante minimale, on va d'abord construire un arbre minimal de Steiner dans un graphe construit à partir de l'arbre d'origine. Pour résoudre le problème de Steiner, nous proposons d'utiliser un algorithme d'énumération permettant la construction de l'arbre couvrant minimal selon un temps qui, si le nombre de feuilles dans l'arbre d'origine est limité, sera polynomial.

Simplifions les notations et designons par $T = (V,E)$ un *arbre filtré* au sens de la définition 3.19 (les feuilles sont des éléments de τ et il n'existe que des nœuds significatifs).

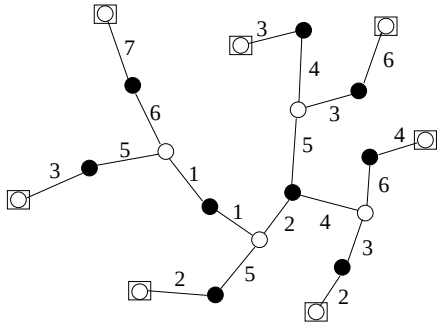
Pour éliminer les circuits introduits par l'union, un problème équivalent peut être formulé (cf. [6]). Si l'on contracte les feuilles de l'arbre T en un seul nœud, on obtient un graphe G' qui contient des circuits. Par la suite, ce nœud sera noté τ_f. La contraction des feuilles de l'arbre présenté sur la figure $7/c$ est illustrée par la figure 8. Si le nombre des feuilles



a)



b)



c)

FIG. 7 – *Simplification d'un arbre d'une quasi-partition*

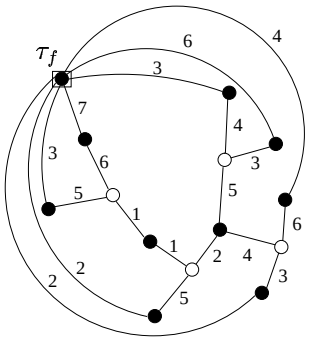


FIG. 8 – *La contraction des feuilles de l'arbre*

contractées est égal à k, la contraction introduit $(k-1)$ cycles indépendants dans G'. La recherche d'une forêt joignante minimale dans l'arbre filtré T correspond à la recherche d'un arbre de Steiner minimal dans le graphe G' ([6]).

Remarque : Au cours de la transformation, les feuilles de T sont regroupées, contractées en un seul nœud et ce nœud appartient à l'ensemble des nœuds à couvrir dans le graphe G'. C'est pourquoi, seuls les nœuds de R différents des feuilles de T sont vraiment significatifs pour calculer l'arbre minimal de Steiner.

3.3.1 Rappel sur des algorithmes de construction de l'arbre minimal de Steiner

Pour construire l'arbre minimal de Steiner dans notre cas particulier présenté au paragraphe précédent, les algorithmes d'énumération d'arbres couvrants (Spanning Tree Enumeration Algorithms) nous semblent bien adaptés. Le premier algorithme d'énumération d'arbres couvrants a été proposé par Hakimi (cf. [7] et [9]). Dans cette classe d'algorithmes, on énumère les ensembles W de nœuds qui contiennent les nœuds du groupe R à couvrir du graphe $G = (V,E)$, i.e.,:

$$R \subseteq W \subseteq V$$

Pour chaque ensemble W tel que le sous-graphe $G_W = (W,E_W)$ engendré par W soit connexe, on calcule l'arbre couvrant minimal $MST(W)$ dans le sous-graphe G_W. Finalement, on choisi l'optimum W^* réalisant la longueur minimale parmi les arbres calculés. Si $mst(W)$ indique la longueur de l'arbre $MST(W)$:

$$mst(W^*) = \min_{R \subseteq W \subseteq V} mst(W).$$

Dans le graphe G, appelons *chemin élémentaire* un chemin dont seules les extrémités appartiennent soit à R, soit à l'ensemble $\mathcal{S}$ des S-nœuds. Notons qu'un arbre minimal de

Steiner de R se compose de chemins élémentaires. Rappelons ici qu'un graphe G' transformé d'un arbre filtré T ne contient que des nœuds significatifs, appartenant à R et/ou à $\mathcal{S}$.

Lawler a prouvé que le calcul de W^* dans G est équivalent au calcul d'un arbre couvrant minimal de R dans la fermeture métrique du graphe G (toujours par énumération) [8]. Sachant que chaque chemin élémentaire de $MST(W^*)$ correspond à une arête de la fermeture métrique, il est sûr que l'image $MST'(W^*)$ de $MST(W^*)$ dans la fermeture est aussi un arbre couvrant de longeur minimal parmi les arbres $MST'(W)$ possibles qui couvrent R dans la fermeture. Lawler a également constaté que le calcul dans la fermeture métrique est plus avantageux que celui effectué sur le graphe d'origine G si G est dense et si l'ensemble R a une cardinalité beaucoup plus faible que l'ensemble des nœuds du graphe G, i.e., si :

$$2r - 2 < v, \tag{2}$$

où r et v désignent respectivement les cardinalités de R et de V.

L'algorithme proposé par Hakimi et par Lawler calcule l'arbre minimum de Steiner selon une complexité temporelle $O(v^2\dot{2}^{v-r} + v^3)$ (cf. [11]). Le calcul étant effectué sur la fermeture métrique ou sur le graphe d'origine selon que l'inégalité 2 est vérifiée ou non, mais l'expression de la complexité reste la même). Dans cette expression, le membre v^3 correspond au calcul des distances sur le graphe G, en utilisant l'algorithme bien connu de Floyd.

Par ailleurs, des méthodes de réduction du problème de Steiner et un algorithme original de génération d'arbres minimaux de Steiner ont été publiés par Balakrishnan et Patel (cf. [10]). L'algorithme de Balakrishnan et de Patel peut être considéré comme une version de l'énumération d'arbres couvrants et peut être résumé comme ci-après. Dans un premier temps, les auteurs proposent des réductions des nœuds et des arêtes du problème. Vue leur importance, nous parcourons ces propositions dans la sous-section 3.3.2.

Supposons maintenant, que le graphe G du problème ne peut plus être réduit. Un nœud artificiel a est ajouté au graphe G d'origine et ce nœud est connecté à tous les nœuds de branchement de G n'appartenant pas à R (ensemble $\mathcal{S}$) ainsi qu'à un nœud arbitraire b du groupe R à couvrir. Toutes ces nouvelles arêtes sont de longueur nulle. Dans le graphe $\tilde{G}$ ainsi modifié, on cherche l'arbre couvrant minimal $\tilde{T_R}$ (qui couvre tous les nœuds de $\tilde{G}$) vérifiant le critère de Balakrishnan et Patel (BP) suivant : si l'on enlève l'arête (a,b) de l'arbre, il reste un sous-arbre qui couvre tous les nœuds de R.
Dans ce cas, après avoir supprimé l'arête (a,b), le sous-arbre qui couvre R est un arbre minimal de Steiner de R. Notons une propriété intéressante d'un tel arbre$\tilde{T_R}$: si l'une des arêtes ajoutées différente de (a,b) appartient à $\tilde{T_R}$, alors l'extrémité différente de a de cette arête a un degré 1 dans $\tilde{T_R}$ (cette extrémité est une feuille de $\tilde{T_R}$) ; car sinon cette extrémité différente de a serait reliée à un nœud de R sans passer par a et le critère BP ne serait pas vérifié. Notons ici, que cette extrémité est un des nœuds de branchement du graphe.

Remarque : Il est évident qu'un nœud de branchement appartenant à S et à R à la fois ne peut pas devenir feuille dans l'arbre couvrant $\tilde{T_R}$. Si un tel nœud était feuille dans $\tilde{T_R}$, il resterait coupé des autres nœuds de R après la suppression de l'arête (a,b).

En construisant les arbres couvrants de $\tilde{G}$ dans l'ordre croissant des longueurs, le premier arbre qui correspond au critère BP ci-dessus est la solution. Pour générer la suite des arbres couvrants, les auteurs proposent d'utiliser l'algorithme de Gabow [12]. Cette méthode fournit les arbres dans l'ordre non décroissant et selon une complexité temporelle $O(n|E|)$ pour n arbres. Afin d'accélérer la recherche, Balakrishnan et Patel utilisent une version modifiée de l'énumération de Gabow où plusieurs réductions et restrictions sont appliquées (cf. la section 3.3.2). Malgré l'accélération, la faiblesse de l'approche de Balakrishnan et de Patel provient de ce qu'il n'existe pas de garantie sur une obtention "relativement rapide" de la solution au cours de l'énumération des arbres. Les arbres couvrants d'un graphe étant nombreux, la détermination du premier arbre satisfaisant le critère BP peut être précédé par l'examen d'un grand nombre d'arbres (examen de tous dans le pire cas).

3.3.2 Réductions proposées dans l'approche de Balakrishnan et Patel

Dans leur article (cf. [10]), Balakrishnan et Patel proposent différentes possibilités de réduction du problème de Steiner dans le cas d'un graphe quelconque. Dans le but de réduire encore notre graphe G' transformé d'un arbre filtré à l'aide de l'union des feuilles de ce dernier, énumérons ces réductions potentielles.

Pour faciliter la description de ce problème de Steiner, considérons le graphe $G = (V,E)$ et partitionnons les nœuds et les arêtes de la façon suivante :

- un nœud est dit de type R s'il appartient à l'ensemble R de nœuds à couvrir,
- un nœud est dit de type S s'il appartient à l'ensemble $S = V \backslash R$ (ne pas confondre avec l'ensemble $\mathcal{S}$ de nœuds de branchement),
- l'arête $(i,j) \in E$ est dite
  - une arête R-R, si $i \in R$ et $j \in R$,
  - ou une arête S-S, si $i \in S$ et $j \in S$,
  - ou une arête R-S, si une de ses extrémités appartient à R et l'autre à S.

Les premières réductions potentielles de l'article cité sont basées sur l'étude d'arbres minimaux de Steiner et constatent les simples propriétés suivantes :

i) Les feuilles de type S de G sont sans conséquence sur la solution optimale et peuvent être éliminées.
   Dans notre cas, de par la construction de notre graphe transformé G', celui-ci ne contient pas de feuille de type S.

ii) Les chemins d'extrémités de type R qui ne contiennent que des nœuds internes de type S et de degré deux peuvent être remplacés par une seule arête R-R. Cette arête doit avoir la même longueur que le chemin remplacé.
   Rappelons que la réduction de niveau 3 (cf. définition 3.18) utilisée sur notre graphe transformé G' est équivalente à cette opération.

iii) Dans l'arbre couvrant recherché, les feuilles de type R sont obligatoirement rattachées à leur voisin. En conséquence, si le graphe G contient des feuilles de type R, ces feuilles peuvent être aggrégées avec leur nœud voisin.

Notons que la réduction de niveau 1 proposée au paragraphe 3.2 remplace les branches ayant une feuille de type R par leur point d'attache. En conséquence, il n'existe pas de feuille de type R dans notre graphe G' construit à partir l'arbre filtré T.

iv) Si, pour deux nœuds adjacents, il existe un chemin plus court que l'arête qui les connecte, celle dernière peut être éliminée du graphe G.
Signalons ici que nous allons utiliser cette réduction par la suite sur notre graphe G' puisque G' n'est pas toujours métrique.

Balakrishnan et Patel analysent également les liaisons entre les différents arbres couvrants associés à différents sous-graphes de G et proposent les réductions suivantes.

Proposition BP_I : agrégation de nœuds. Soit $S^* \subset S$ le sous-ensemble de nœuds de type S qui appartient à la solution. Si les deux extrémités d'une arête appartiennent à $R \cup S^*$ et que l'arête fait partie de l'arbre $MST(G)$ qui est l'arbre minimal couvrant de G, alors l'arête en question appartient aussi à l'arbre minimal de Steiner de R.

Autrement dit : l'arbre minimal de Steiner doit contenir toutes les arêtes du $MST(G)$ qui connectent deux nœuds de $R \cup S^*$.

Remarque : Naturellement, on ne connaît pas par avance l'ensemble S^* ; toutefois, cette proposition reste utile.

Agrégation des nœuds de type R : D'après la proposition BP_I, chaque arête R-R de $MST(G)$ appartient aussi à l'arbre minimal de Steiner. En conséquence : si deux nœuds de type R sont adjacents dans $MST(G)$, on peut les regrouper en un seul nœud de type R pour obtenir un problème équivalent.

Balakrishnan et Patel indiquent également que le regroupement des nœuds peut introduire des arêtes parallèles. Dans ce cas, pour retrouver un 1-graphe équivalent, seule l'arête la plus courte parmi les arêtes parallèles doit être gardée.

Restriction d'arêtes R-S : Supposons que l'arbre $MST(G)$ contient une arête (i,j) de type R-S. Soit $i \in S$ et $j \in R$. Selon la proposition BP_I, l'arbre minimal de Steiner contient (i,j), ssi $i \in S^*$. Si la valeur 1 d'une variable booléenne y_{ij} indique l'appartenance de l'arête (i,j) à l'arbre minimal de Steiner, alors cette restriction correspond à la contrainte suivante : $y_{ij} \geq y_{ik}$, $\forall (i,k) \in E$. Les contraintes établies à partir de $MST(G)$ peuvent être utilisées dans certaines méthodes de construction de l'arbre minimal de Steiner, par exemple dans la méthode de relaxation lagrangienne. Notons que notre algorithme de construction ne profite pas de ces restrictions.

Proposition BP_{II} : suppression d'arêtes R-R. Soit $MST(R)$ l'arbre couvrant minimal du graphe engendré par R. L'arbre minimal de Steiner ne contient pas d'arête R-R n'appartenant pas à $MST(R)$.

Si le graphe engendré par R est connexe, seules les $(r-1)$ arêtes de ce graphe appartenant à $MST(R)$ doivent être gardées pour la suite. Cette proposition est illustrée figure 9 où des arêtes R-R en pointillé, de longueur 4 et 6, sont inutiles pour la suite du calcul. Balakrishnan et Patel indiquent que dans un graphe complet, cette suppression réduit le nombre des arêtes R-R par le facteur $r/2$.

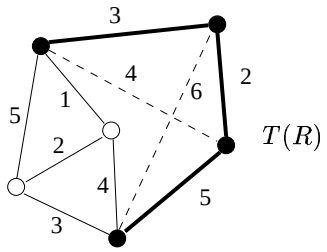


FIG. 9 – *Suppression d'arêtes R-R*

Proposition BP_{III} : suppression d'arêtes R-S. Une arête (i,j) de type R-S de G avec $i \in S$ et $j \in R$ peut appartenir à l'arbre minimal de Steiner de R sur G, ssi l'arbre couvrant minimal $MST(R \cup \{i\})$ du graphe engendré par $(R \cup \{i\})$ contient cette arête.

Propositions BP_{IV} : suppression d'arêtes S-S. Une arête (i,j) de type S-S de G peut appartenir à l'arbre minimal de Steiner de R sur G, ssi l'arbre couvrant minimal $MST(R \cup \{i\} \cup \{j\})$ contient cette arête.

Remarque : La construction d'un arbre couvrant minimal d'un graphe a une complexité d'ordre quadratique ; ce qui explique que les propositions BP_{III} et BP_{IV} ne sont pas vraiment intéressantes pour un filtrage préliminaire. Par contre, ces propositions peuvent être utilisées au fur et à mesure de l'exécution d'un algorithme d'énumération d'arbres couvrants pour enlever une arête du graphe dès qu'on sait (grâce aux propositions BP_{III} et BP_{IV}) qu'elle n'appartient pas à la solution optimale.

3.3.3 Construction de la forêt joignante minimale

Nous montrons maintenant que, pour notre cas particulier, le calcul de l'arbre optimal de Steiner est plus avantageux sur le graphe d'origine et nous présentons un algorithme de construction de cet arbre. Si l'on peut supposer que le nombre de feuilles est toujours limité par une valeur fixée, l'algorithme présenté est de complexité temporelle polynomiale.

Soit R' l'ensemble des nœuds à couvrir dans le graphe G' créé à partir d'un arbre filtré (la figure 8 illustre un tel graphe). Soit r' la cardinalité de R' (rappelons que R' contient aussi le nœud unifié τ_f). Rappelons que $r = |R|$. En notant h le nombre de nœuds de R qui sont aussi des feuilles de l'arbre filtré T, on a $r' = r - h + 1$. Soit k le nombre de nœuds de type S de l'arbre T (les S-nœuds n'appartenant pas à R). Rappelons que la complexité d'un algorithme d'énumération d'arbre dans la fermeture métrique est en $O(2^k(r-h+k+1)^2)$ ou encore $O(2^k(r'+k)^2)$, sans le calcul des distances.

Notre algorithme construisant l'arbre minimal de Steiner sur le graphe d'origine est aussi du type énumération d'arbres couvrants mais effectue la recherche sur un ensemble restreint.

En suivant la proposition de Balakrishnan et Patel, définissons un graphe modifié G'' obtenu à partir de G' de la façon suivante.

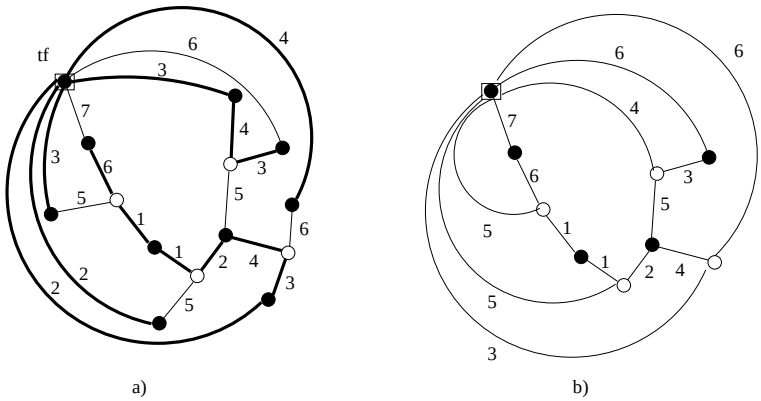


FIG. 10 – *Agrégation de nœuds R*

Dans un premier temps, on calcule $MST(G')$ et l'on effectue les réductions possibles (agrégation de nœuds de type R) selon la proposition BP_I de Balakrishnan et Patel. Soit R'' l'ensemble réduit de nœuds de type R.

Si le graphe engendré par R'' est connexe, on calcule ensuite $MST(R'')$ pour effectuer les réductions possibles selon la proposition BP_{II} (suppression d'arêtes R-R). Le lecteur pourra vérifier que le nombre de composantes connexes du graphe engendré par R'' est égale au nombre de composantes connexes du graphe engendré par R'.

Dans le pire des cas, l'agrégation de nœuds de type R n'est pas possible et le nombre de nœuds du graphe ne change pas. Néanmoins, la cardinalité de R'' est toujours bornée par $r' \leq r + 1$. Dans la suite, nous calculons toujours la complexité des algorithmes pour le cas pessimiste.

La figure 10 illustre certaines réductions. Nous y reprenons le graphe G' de la figure 8 car l'agrégation d'un sous-ensemble de nœuds de type R est possible. La figure $10/a$ indique l'arbre couvrant minimal du graphe entier $MST(G')$. L'agrégation est basée sur les arêtes R-R appartenant à cet arbre ; le résultat de l'agrégation est présenté sur la figure $10/b$. Dans cet exemple, le graphe engendré par R'' (ou par R') n'est pas connexe et la suppression d'arêtes R-R n'est pas possible.

Après les réductions et toujours pour créer le graphe G'', ajoutons un nœud artificiel a connecté à tous les nœuds de type S par des arêtes artificielles de longueur nulle ; choisissons le nœud τ_f (ou l'agrégation de nœuds qui le contient) comme nœud b. Notons que le nœud τ_f (simple ou agrégé par BP_I) appartient toujours à l'ensemble des nœuds à couvrir. Le résultat de ces transformations est le graphe G''.

Dans G'', appelons O le sous-graphe engendré par les arêtes artificielles (ce sous-graphe sera toujours un arbre de longueur nulle).

Notre proposition prend en compte la constatation de Balakrishnan et Patel concernant l'appartenance des arêtes artificielles à la solution. Pour énumérer les cas possibles, nous

allons énumérer les sous-graphes compatibles de O. Si l'une des arêtes artificielles appartient à T''_R (cf. la section 3.3.1), alors cette appartenance introduit des contraintes, des inhibitions pour la suite de la construction. L'extrémité S-nœud d'une telle arête étant obligatoirement une feuille dans la solution, les arêtes adjacentes de ce nœud ne peuvent pas être sélectionnées et doivent être inhibées pendant la construction.

L'algorithme et le lemme suivant nous permettent de déterminer l'arbre minimal de Steiner de R' dans G' :

Algorithme $\mathcal{A}_1$:
début
$Lgmin = +\infty$
à partir de G', déterminer le graphe réduit et modifié G''
pour tout sous-arbre $O_i \subseteq O$ **faire**:
calculer le nouveau graphe G''_{O_i} obtenu en supprimant :
i) les arêtes (de longueur non nulle) de G'' adjacentes des feuilles de O_i excepté la feuille b,
ii) les arêtes de longueur nulle n'appartenant à O_i ;
si G''_{O_i} est connexe, **alors**
calculer l'arbre minimal couvrant de G''_{O_i} correspondant T_{O_i}
si $d(T_{O_i}) < Lgmin$, **alors**
$T^*_O := T_{O_i}$;
$Lgmin := d(T_{O_i})$;
finsi;
enlever les arêtes exclues selon les propositions BP_{III} et BP_{IV};
finsi;
fait;
fin

Remarque : L'agrégation des nœuds de type R n'augment pas la complexité de l'algorithme et peut être faite lors du premier parcours de la boucle principale de l'algorithme si l'on choisit pour O_1 le sous-arbre ne contenant que l'arête (a,b). Dans ce cas, l'arbre calculé T_{O_1} correspond à l'arbre couvrant minimal $MST(G')$ et si elle est possible, l'agrégation est réalisable sans surcoût.

Lemme 3.20 *L'arbre minimal T^*_O obtenu par l'algorithme $\mathcal{A}_1$ contient un arbre minimal de Steiner de R''. Il est obtenu en enlevant de T^*_O le sous-arbre de longueur nulle O^* qu'il contient.*

Preuve.
Soit $\mathcal{E}$ l'ensemble des arbres couvrants de G'' tels que, si une arête de longueur nulle différente de (a,b) appartient à l'arbre, alors son extrémité opposée à a est une feuille.

Il suffit de montrer que l'arbre de longueur minimale de l'ensemble $\mathcal{E}$ est l'arbre T^*_O sélectionné comme ci-dessus.

Si l'on enlève l'arête (a,b) de T_O^*, on obtient au plus deux composantes connexes. Une des composantes ne contient que des arêtes de longueur nulle et de cette façon ne couvre pas de nœuds de R''. Cette composante peut être vide.

Par contre, l'autre composante T_R^* (qui est un arbre, puisque T_O^* est un arbre) couvre tous les nœuds de R''. En plus, $d(T_R^*) = d(T_O^*)$ puisque les arêtes n'appartenant pas à T_R^* sont toutes de longueur nulle. Supposons que l'arbre minimal de Steiner de R'' ne soit pas T_R^* mais l'arbre T_{R_i}, c-à-d : $d(T_{R_i}) < d(T_R^*)$. Ce dernier est le sous-arbre d'un arbre couvrant T_{O_i} différent de T_O^*, et en suivant le même raisonnement : $d(T_{R_i}) = d(T_{O_i})$. La contradiction est évidente, T_R^* ne peut être qu'un arbre minimal de Steiner de R''. ■

En utilisant la notation introduite au début de cette section 3.3.3, le lemme suivant fournit une borne supérieure de la complexité de l'algorithme $\mathcal{A}_1$.

Lemme 3.21 *La complexité de l'algorithme $\mathcal{A}_1$ est bornée par $O(2^k(r-h+k+1)^2)$.*

Preuve. Supposons qu'il n'existe pas de simplification possible : $R'' = R'$.

Le nombre des nœuds de type S étant égal à k, on peut énumérer 2^k sous-arbres O_i différents.

Le graphe G' contient $(r-h+1+k)$ nœuds. Pour un sous-arbre O_i donné, la construction de l'arbre couvrant minimal se fait sur le graphe G''_{O_i} contenant moins de $(r-h+1+k)$ nœuds ; sa complexité est donc bornée par $O(r-h+1+k)^2$. Dans le pire des cas, tous les graphes G''_{O_i} sont connexes et il faut calculer l'arbre couvrant minimal des 2^k arbres différents de longueur nulle. Dans ce cas pessimiste, la complexité de l'algorithme est bornée par $O(2^k(r-h+k+1)^2)$.

■

Remarque : Comme les observations de Lawler le laissaient prévoir, le calcul de l'arbre minimal de Steiner dans le graphe d'origine peut être plus avantageux que dans la fermeture métrique. On espère aussi que notre algorithme $\mathcal{A}_1$ bénéficie des avantages qui découlent des propriétés topologiques du graphe d'origine du problème, propriétés qui sont perdues si l'on utilise la fermeture. Une telle propriété est évidente en utilisant l'algorithme proposé : si un sous-arbre O_i provoque une coupe de G'', le calcul de l'arbre couvrant minimal du graphe G''_{O_i} n'est pas effectué. En utilisant la matrice d'incidence sommet-sommet des graphes, la détection de la connexité d'un graphe ayant v arêtes nécessite un temps de calcul proportionnel à v. De cette façon, la détection des sous-arbres O_i provoquant une coupe de G'' coûte moins cher (en temps de calcul) que le calcul des arbres couvrants des graphes G''_{O_i} correspondants. En général, l'algorithme $\mathcal{A}_1$ devrait être efficace car le nombre de sous-arbres explorés devrait être en général bien inférieur au terme 2^k exprimé par le lemme. Un autre avantage de l'algorithme vient des éventuelles réductions puisque, bien évidemment, si l'on peut diminuer le nombre de nœuds du problème, alors le temps de calcul décroit.

La figure 11 contient un exemple illustrant le mécanisme de l'algorithme $\mathcal{A}_1$ proposé. Supposons que l'agrégation des nœuds soit réalisée comme montré figure 10. La figure $11/a$ illustre le choix des nœuds a et b ainsi que l'ajout des arêtes de longueur nulle. L'arbre en pointillé correspond au sous-graphe O de G''. Les figures $11/b$ - $11/l$ présentent les cas

où les graphes G''_{O_i} restent connexes. Sur ces figures, les sous-arbres O_i sont en pointillé gras (les sous-arbres O_i incompatibles ne sont pas énumérées ici). Ces figures $11/b$ - $11/l$ indiquent en gras les arbres couvrants minimaux correspondant aux différentes inhibitions. La solution optimale est l'arbre de longueur minimale parmi ces arbres. Cet exemple possède deux solutions : les arbres des figures $11/b$ et $11/g$.

Après avoir calculé l'arbre minimal de Steiner de l'ensemble R'', il faut recréer celui de l'ensemble R'. Cette création nécessite un simple ajout des arêtes R-R enlevées par l'agrégation de nœuds de type R. Dans le cas de l'exemple numérique utilisé dans ce paragraphe, l'ajout nécessaire est illustré par la figure 12.

Ayant calculé l'arbre de Steiner optimal T'_R sur G', on cherche maintenant à obtenir la forêt déductible maximale de l'arbre filtré T à l'aide du lemme suivant.

Lemme 3.22 *L'arbre minimal de Steiner T'_R de $R' = R \bigcup \tau_f$ dans G' correspond à une forêt joignante partielle minimale F_j dans T. En conséquence : la forêt déductible maximale F_d de T relative aux feuilles se compose des sous-arbres qui n'appartiennent pas à F_j. Autrement dit :*

$$F_d = T \ominus F_j$$

Preuve.

i) Dans le graphe G', l'arbre T'_R couvre R et le nœud τ_f. Chaque sous-arbre de racine τ_f est un arbre couvrant minimal du nœud τ_f et d'un sous-ensemble de nœuds de R. Evidemment, tous les nœuds de R sont couverts par les sous-arbres de τ_f. La forêt F_j qui correspond à T'_R dans T se compose des sous-arbres de τ_f et est une forêt joignante minimale relativement à τ.

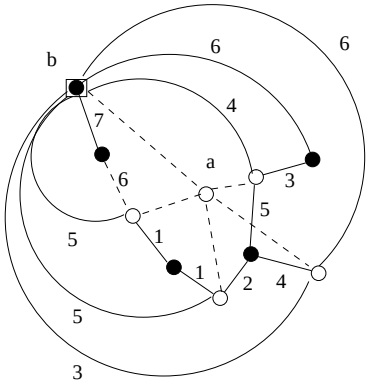
ii) Si l'on enlève une forêt joignante minimale d'un arbre, on obtient la forêt complémentaire, i.e., une forêt déductible maximale (cf. la relation 1). ■

Si l'arbre couvrant de départ du paragraphe 3.2.2 a été simplifié en utilisant les réductions des niveaux 1 et 2, alors il faut encore réconstruire la forêt déductible maximale du problème d'origine. Cette reconstruction est possible en deux étapes :

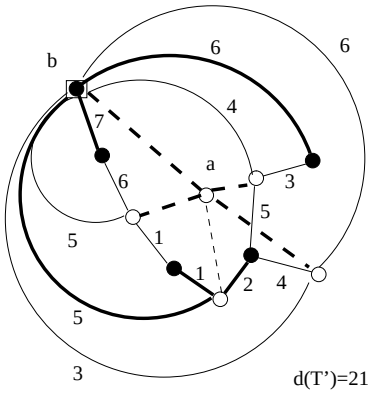
1) A partir de la forêt joignante minimale du problème réduit, on peut créer la forêt joignante minimale du problème d'origine, en ajoutant les sous-arbres qui ont été réduits en leurs points d'attache selon la réduction de niveau 1.
2) La forêt déductible maximale dans l'arbre d'origine est la forêt complémentaire de la forêt joignante minimale du problème d'origine.

Cette démarche est illustrée figure 13 où la sous-figure $13/a$ présente la forêt joignante minimale du problème réduit correspondant à la solution de la sous-figure $11/b$. La figure $13/b$ montre comment cette forêt peut être complétée pour obtenir la forêt joignante minimale du problème d'origine. La forêt déductible maximale de l'arbre d'origine obtenue par l'opération de soustraction minimale est représentée figure $13/c$.

Remarque : L'exemple de cette sous-section permet d'illustrer le gain en temps calcul que l'on peut espérer en utilisant l'algorithme $\mathcal{A}_1$. En négligeant le gain obtenu par la technique de décomposition (par la partition de l'arbre d'origine), prenons comme point de départ un

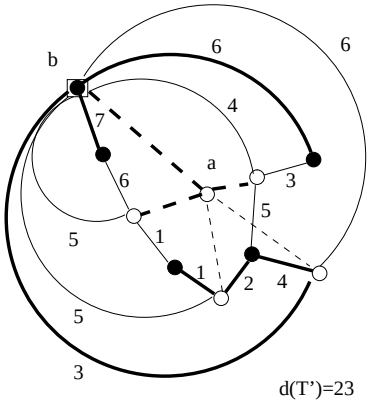


a)



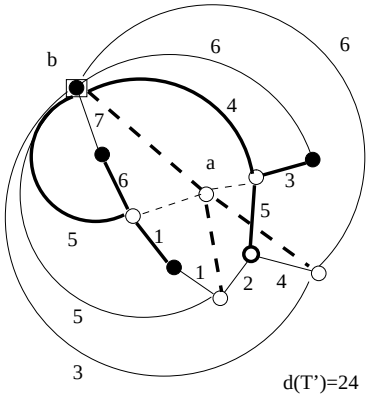
d(T')=21

b)



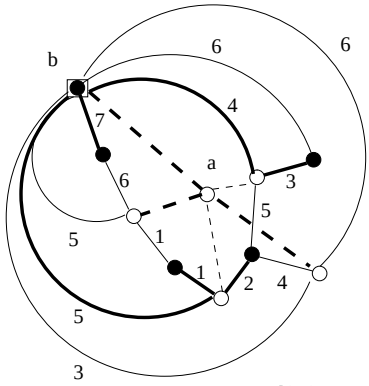
d(T')=23

c)



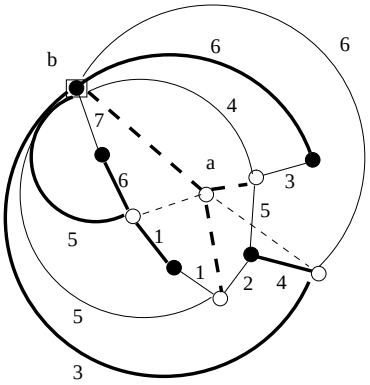
d(T')=24

d)



d(T')=22

e)



d(T')=25

f)

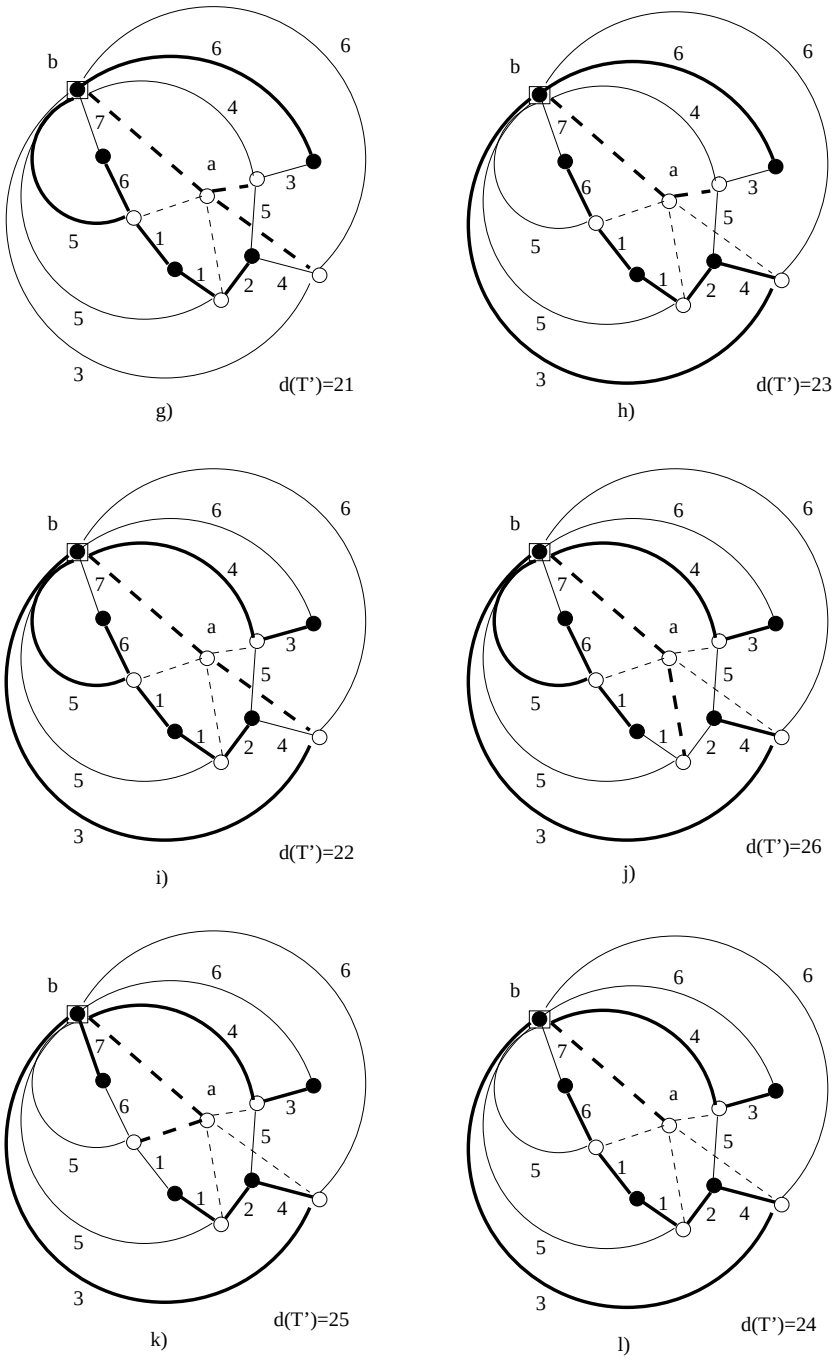


FIG. 11 – *Le calcul proposé de l'arbre minimal de Steiner*

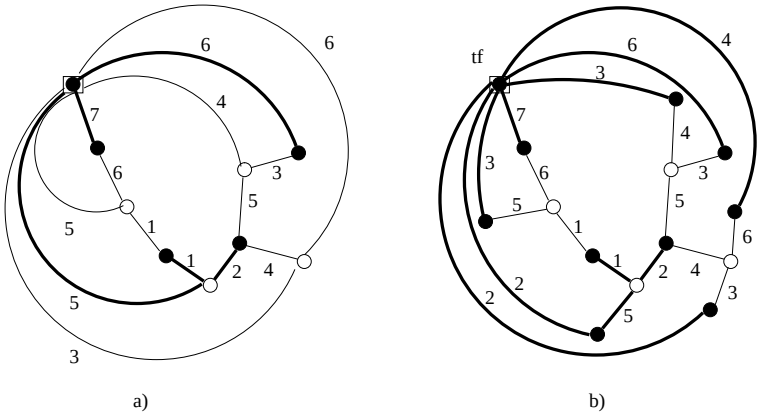


FIG. 12 – *Réintroduction des arêtes $R-R$ enlevées*

unique élément de la quasi-partition, l'élément illustré figure $7/a$. Pour cet arbre, il y a 17 nœuds à couvrir. En unifiant les feuilles de cet arbre, on obtient un graphe dans lequel la recherche de l'arbre minimal de Steiner correspond à la construction de la forêt joignante minimale sur l'arbre. Puisque l'arbre contient 14 nœuds internes à couvrir et qu'il faut couvrir le nœud unifié, le nombre de nœuds dans le groupe à couvrir est $r'_a = 15$. Dans l'arbre, le nombre de nœuds de branchement n'appartenant pas à l'ensemble R'_a à couvrir est $k_a = 10$. Si l'on applique l'algorithme de l'énumération d'arbres couvrants directement à ce problème, le temps de calcul est à $O(2^{10}\dot{2}5^2)$.

Le résultat de la réduction de niveau 1 illustré sur la figure $7/b$ nous conduit à un graphe contenant $r'_b = 9$ nœuds internes à couvrir et $k_b = 8$ nœuds de branchement n'appartenant pas à R'_b. Après la simplification de niveau 2 (arbre illustré figure $7/c$) et après unification des feuilles de l'arbre, on obtient le graphe présenté sur la figure 8, où $r'_c = 10$ et $k_c = 4$. Les méthodes de Balakrishnan et de Patel réduisent également la taille du problème de l'exemple. Ici, selon le témoignage de la figure 10, l'agrégation de certains nœuds de type R est réalisable. Dans le graphe réduit, il y a $r'_d = 5$ nœuds de type R et $k_d = 4$ nœuds de type S. Comme la figure 11 le montre aussi, l'algorithme $\mathcal{A}_1$ ne calcule que 11 arbres couvrants au lieu de $2^4 = 16$ arbres, en revanche, pour les 16 cas possibles, il faut vérifier la connexité des graphes. Au cours de la construction de l'arbre minimal de Steiner et en conséquence de la forêt déductible maximale du sous-arbre mentionné, le gain peut finalement être caractérisé par le rapport $(2^{10} \cdot 25^2)/(11 \cdot 9^2 + 16 \cdot 9) = 618$.

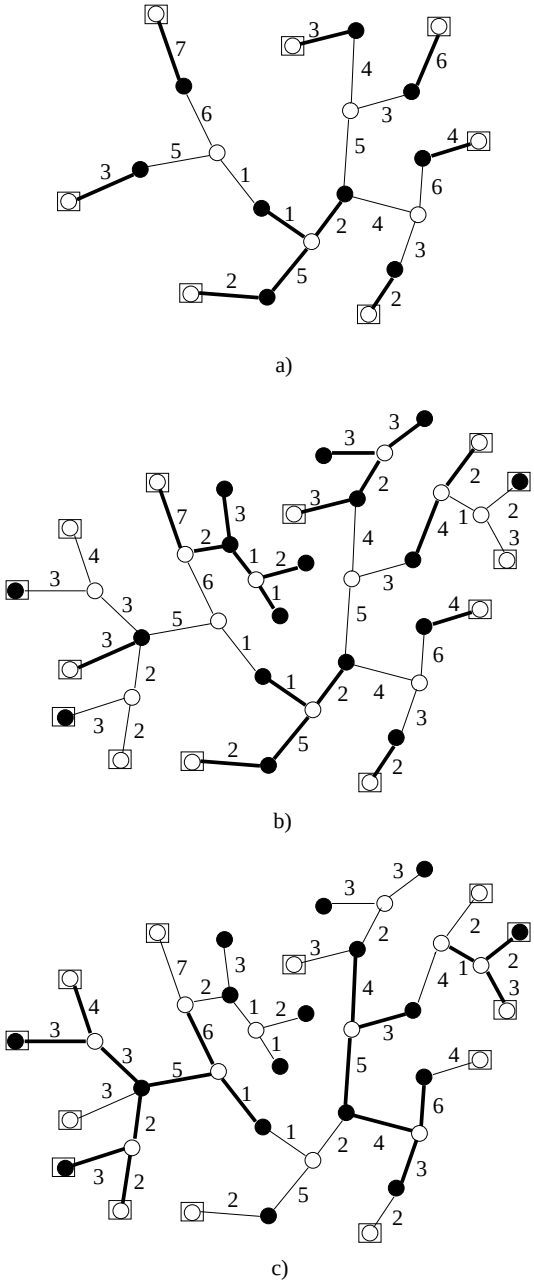


FIG. 13 – *La construction de la forêt déductible maximale de l'exemple*

4 Algorithme de détermination d'une forêt déductible maximale

Soit l'arbre valué $T_R = (V_R, E_R, d)$, le groupe R à couvrir et le groupe τ des connexions qui touchent l'arbre donnés en entrée de l'algorithme. On souhaite déterminer la forêt déductible de T_R relativement à τ. Pour présenter les algorithmes plus aisement dans la suite, supposons l'existance des types **graphe**, **arbre**, **forêt**, **partition** et **groupe** (ce dernier pour référencer un ensemble de nœuds).

4.1 Algorithme principal

L'algorithme suivant indique les grands pas des simplifications décrites dans le paragraphe 3.2, identifie et retourne la forêt déductible selon les lemmes du paragraphe 3.3. Les fonctions appelées par cet algorithme sont détaillées et analysées dans la suite. La figure 14 résume l'enchaînement des traitements au cours de cet algorithme.

forêt FDM (arbre T_R, groupe R, groupe τ)
1 partition P = quasi-partition(T_R, τ) ;
2 P = filtrage(P, τ) ; /* suppression des composants sans circuit */
3 forêt Res $= \emptyset$; $E = \emptyset$; $D = \emptyset$;
4 Pour $\forall T \in P$ faire :
5 arbre $\tilde{T}$ = réduction($T, \tau, R; E; D$) ; /* reduction de niveaux 1 et 2*/
6 graphe G' = contraction($\tilde{T}, \tau$) ; /* regroupement des feuilles */
7 arbre $T_{\mathtt{St}}$ = STABP(G', R) ; /* arbre optimal de Steiner */
8 forêt F_j = joint($T_{\mathtt{St}}, \tau$) ; /* forêt joignante minimale correspondante */
9 forêt $F_{\mathtt{ded}} = T \ominus F_{\mathtt{jm}} \bigcup D$; /* forêt déductible maximale du composant */
10 Res $=$ Res $\bigcup F_{\mathtt{ded}}$;
11 fait ;
12 retourner(Res) ;
fin.

4.2 Fonctions utilisées

4.2.1 Création d'une quasi-partition

La quasi-partition de l'arbre T_R induite par l'ensemble de nœuds τ peut être déterminée par l'algorithme ci-dessous :

partition quasi-partition (arbre T_R, groupe τ)
1 partition $P = T_R$; /* initialisation par l'arbre T_S */
2 Pour $\forall v \in \tau$ faire :
3 Pour $\forall T \in P$ faire :

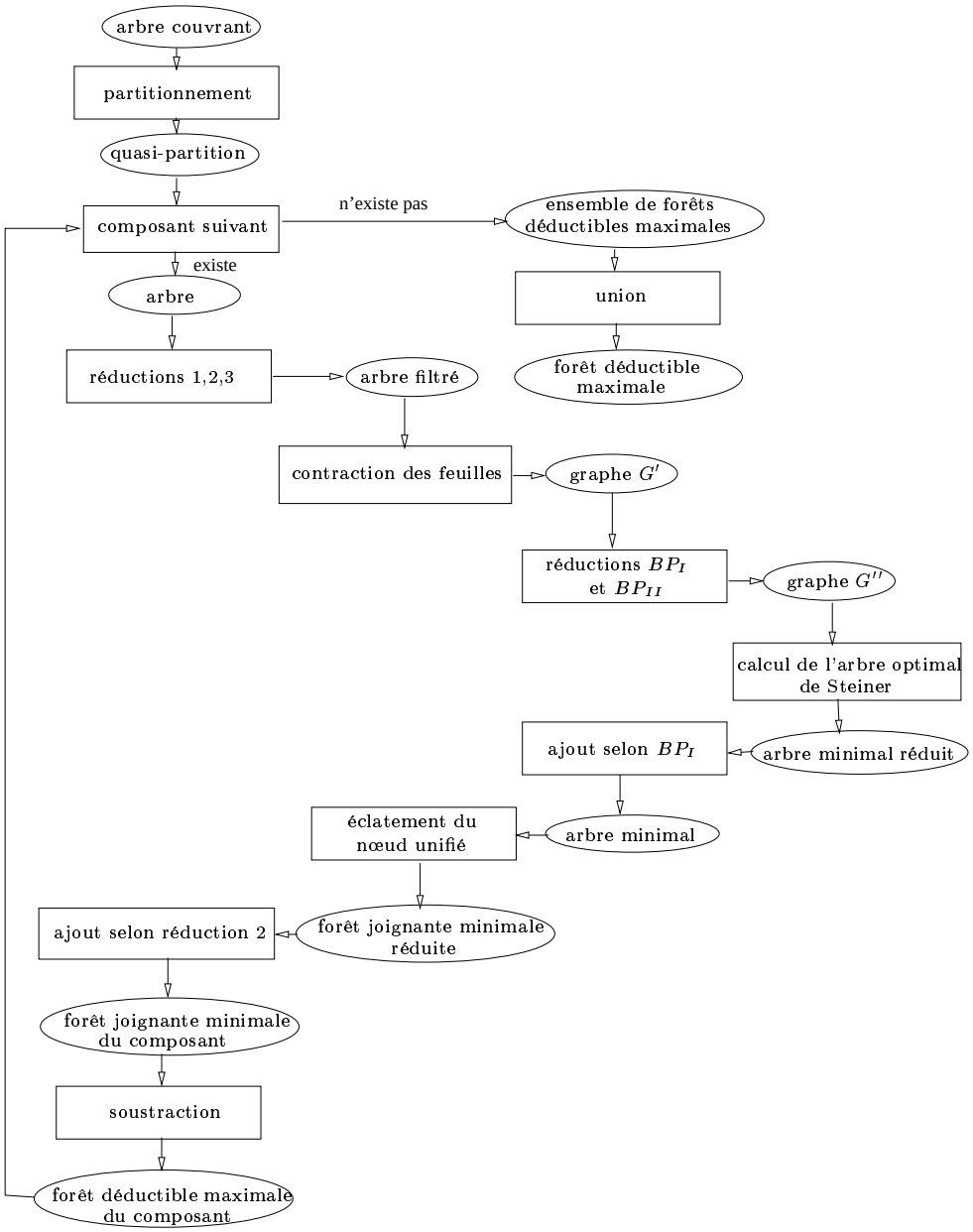


FIG. 14 – *Traitements principaux de l'algorithme proposé*

```
4       si (v ∈ T)
5           alors
6               partition P' = eclater( T,v) ; /* éclater T selon v */
7               P = (P\T) ⋃ P' ; /* remplacer T dans P */
8   fait ;
9  fait ;
10  retourner(P) ;
fin.
```

Pour éclater un arbre selon un de ses nœuds, l'algorithme de la création de la quasi-partition appelle la fonction suivante:

partition eclater(arbre T, nœud v)

```
1  partition P = ∅ ;
2  si (degré(v) = 1)
3    alors P = T ; /* cas d'une feuille */
4    sinon
5       Pour ∀a = arete-adjacente(v) faire :
6           arbre T' = sous-arbre(T,v,a) ; /* sous-arbre contenant a */
7           P = P ⋃ T' ;
8       fait ;
9  retourner(P) ;
fin.
```

Propriété : La complexité de l'algorithme de création de la quasi-partition est en

$$O(|\tau| \cdot |V_S| \cdot \alpha \cdot |V_S|) = O(|\tau| \cdot |V_S|^2)$$

La boucle qui commence à la ligne 2 dans la fonction **quasi-partition** est exécutée $|\tau|$ fois. Dans le pire cas, l'évaluation à la ligne 4 de l'appartenance pour l'ensemble des arbres existant nécessite $|V_S|$ comparaisons. Le coût de l'éclatement d'un sous-arbre peut être majoré par $\alpha \cdot |V_S|$, avec $\alpha < 1$; d'où l'ordre de grandeur de la complexité.

4.2.2 Filtrage des arbres pouvant contenir une forêt déductible

Les arbres de la quasi-partition qui ne contiennent pas au moins deux nœuds de τ ne peuvent pas contenir une forêt déductible. On les enlève de l'ensemble par la fonction:

partition filtrage(partition P, groupe τ)

```
1  partition P' = P ;
2  Pour ∀T ∈ P faire :
3    n(T) = 0 ;
4  Pour ∀v ∈ τ faire :
```

```
5    Pour ∀T ∈ P faire :
6        si (v ∈ T)
7            alors n(T) = n(T) + 1 ;
8    fait ;
9  fait ;
10  Pour ∀T ∈ P faire :
11   si (n(T) < 2)
12       alors P' = P'\T ;
13  fait ;
14  retourner(P') ;
fin.
```

Remarque : Ce filtrage est possible au cours de la création de la quasi-partition et il vaut mieux le faire lors de cette création pour éviter la reprise des arbres de la quasi-partition. Pour chaque arbre créé, le nombre de nœuds appartenant à τ peut être calculé au fur et à mesure. Après le dernier éclatement, on peut immédiatement supprimer les arbres qui ne possèdent pas au moins deux éléments de τ. La définition séparée de ce filtrage n'est présentée ici que par un souci pédagogique. Aussi, pour la complexité de l'algorithme total, on ne tient pas compte de cette opération de filtrage. L'algorithme fournissant la quasi-partition filtrée reste borné par l'expression vue au sous-paragraphe 4.2.1.

4.2.3 Réduction des sous-arbres terminaux

Après n'avoir conservé que les arbres intéressants, on cherche les forêts déductibles maximales de ces arbres. Dans ceux-ci, avant la recherche coûteuse de la forêt, on réalise d'éventuelles réductions de niveau 1, 2 et 3. La fonction suivante retourne l'arbre filtré (cf. définition 3.19). Naturellement, ces réductions ont comme résultat un problème équivalent ; la fonction **réduction** présentée ci-dessous modifie l'arbre T ainsi que des groupes de nœuds τ et R. Notons ici que les sous-arbres enlevés dans la première partie de l'algorithme (par la réduction de niveau 1) ne sont pas déductibles. Par contre, les sous-arbres qui sont remplacés par une branche équivalente dans la deuxième partie de l'algorithme (réduction de niveau 2) appartiennent à la forêt déductible maximale. Il faut les mémoriser à l'aide de la forêt D de la fonction pour un ajout ultérieur (celui-ci se trouve à la fin de l'algorithme principal).

réduction(arbre T, groupe τ, groupe R, forêt E, forêt D)

```
1  nbni = 0; /* niveau 1 */
2  Pour ∀v ∈ R faire :
3    si (degre(v)) = 1) /* feuille de T */
4       alors si ( v ∉ τ) /* feuille dans R pouvant être réduites */
5          { v' = point-d-attache(v); /* premier successeur avec degre(v')>1 */
6          R = (R\v) ⋃ v' ; /* remplacer v par v' */
7          enlever(v,v',T) ; } /* enlever le chemin v-> v' de T */
```

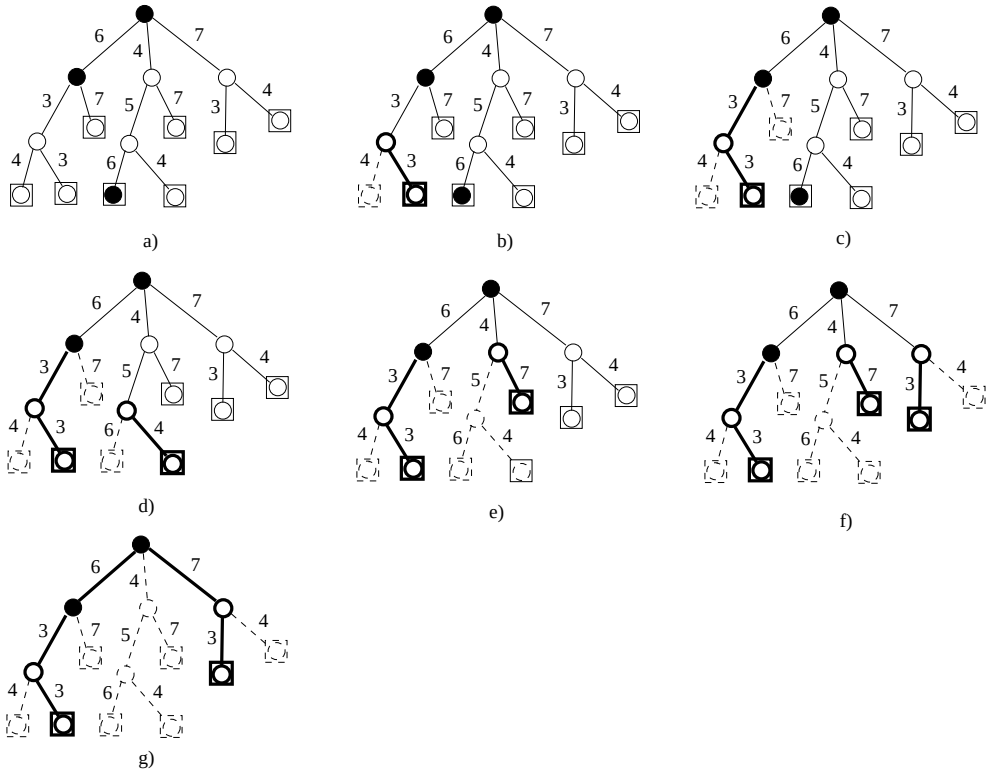


FIG. 15 – *La réduction de niveau 2*

```
8           ajouter(v,v',E) ; } /* ajouter le chemin v-> v' à E */
9    sinon /* nœud interne de T */
10       nbni = nbni + 1 ; /* nombre de nœuds de R qui sont internes */
11       v' = v ;
12  fait ;
13  si (nbni > 0)  /* niveau 2 */
14   remplacer(T,v',R,long,D) ; /* simplification à partir de v' */
fin.
```

Nous proposons la simplification des sous-arbres terminaux (réduction de niveau 2) à l'aide d'une procédure récursive qui réalise le parcours et la simplification de son argument arbre T en un seul passage à partir d'une racine correspondant à un nœud interne de type R de l'arbre T. Si un tel élément n'existe pas, le calcul de la forêt déductible est trivial : l'arbre T entier peut être déduit.

Les règles de simplification d'un arbre de racine v sont les suivantes :

- une feuille ne peut pas être simplifiée ;
- si la racine d'un sous-arbre (le successeur du nœud courant) appartient à R, le sous-arbre est considéré comme un arbre simplifié avec contrainte;
- l'ensemble des sous-arbres simplifiés sans contrainte peut être remplacé par le sous-arbre simplifié le plus court ;
- les sous-arbres simplifiés avec contrainte ne peuvent pas être réduits.

Voici l'algorithme récursif de la simplification.

booleen remplacer(arbre T, nœud v, groupe R, entier $long$, forêt D)

```
1  si (degre(v) = 1) /* cas d'arêt : feuille de T */
2    alors { long = 0 ;
3        return vrai ; }
4  lg = valeurmax ;
5  arbre T_res = ∅ ; /* reconstruction de l'arbre simplifié */
6                    /* à partir de l'arbre vide              */
7  arbre T_1 = ∅ ;
8  booleen rep = vrai ;
9  Pour ∀v' = suc(v) faire : /* successeurs de la racine */
10   arbre T' = sous-arbre (T,v') ; /* sous-arbre de racine v' */
11   forêt D' = ∅ ;
12   booleen rep' = remplacer(T',v',R,long',D') ; /* simplification du sous-arbre */
13   rep = rep & rep' ;
14   si (rep') /* le sous-arbre est simplifié */
15       alors si (long' + d(v,v') < lg) /* sa longueur est minimale */
16           alors
17               long = long' + d(v,v') ;
18               arbre T_1 = T' ⋃ chemin(v,v') ;
19   sinon /* le sous-arbre ne peut pas être simplifié */
20       T_res = T_res ⋃ T' ; /* */
21  fait ;
22  si (lg = valeurmax) /* il y a un sous-arbre réduit */
23   alors T_res = T_res ⋃ T_1 ;
24  D = T ⊖ T_res ;
25  T = T_res ;
26  si (v ∈ R) /*la simplification vers les parents est impossible */
27   alors rep = faux ;
28  retourne(rep) ;
fin.
```

La figure 15 illustre le fonctionnement de cet algorithme et l'évolution de l'arbre simplifié (en gras) ainsi que l'évolution de la forêt déductible trouvée par cette réduction (en pointillé).

Propriété : La complexité de la fonction **réduction** est en $O(|R| + |V_S|)$.

Si β indique la hauteur de l'arbre (c'est aussi une borne supérieure des longueurs des chemins dans l'arbre), alors le coût approximatif de la première partie est à $O(\beta \cdot |R|) = O(|R|)$. La simplification faite par la fonction **remplacer** réalise un parcours complet de l'arbre et peut être caractérisée par $O(|V_S|)$.

4.2.4 Contraction des feuilles de τ

Cette fonction reçoit en entrée un arbre et crée comme résultat un graphe qui correspond à l'arbre en entrée avec la seule différence que les feuilles de l'arbre sont contractées en un seul nœud interne. L'algorithme correspond alors à une simple modification du graphe et dépend de l'implémentation choisie pour représenter les graphes et les arbres.

En tous cas, quelque soit cette implémentation, si l'arbre de départ contient τ' feuilles, il y a τ' modifications à faire. Dans le pire cas, l'arbre contient toutes les feuilles de τ et de cette façon le temps de calcul de la fonction peut être majoré par une valeur proportionnelle à $|\tau|$.

4.2.5 Calcul de l'arbre minimal de Steiner

Pour calculer l'arbre minimal de Steiner, nous utilisons l'algorithme $\mathcal{A}_1$ d'énumération d'arbres couvrants proposé dans la section 3.3.3. Rappelons que, dans cet algorithme pour énumérer les topologies qui sont différentes selon l'appartenance ou non des différents S-nœuds du graphe à l'arbre, nous utilisons la méthode de Balakrishnan et de Patel. Contrairement à leur proposition originale, notre algorithme ne crée pas la suite des arbres couvrants de longueur croissante, mais il calcule l'arbre couvrant minimal de chaque configuration. Plus précisemment : on ne calcule l'arbre couvrant que pour les topologies admissibles. Si l'ensemble des arêtes adjacentes des feuilles du sous-arbre O_i contient une coupe du graphe (c-à-d : il existe deux composants connexes qui ont des nœuds de R), alors l'algorithme passe à la configuration suivante. Pour une configuration donnée, la méthode MSTC calcule l'arbre couvrant minimal sous les contraintes qui correspondent aux inhibitions (aux arêtes adjacentes de O_i). On présente maintenant en détail l'algorithme d'énumération d'arbres couvrants noté sous cette forme détaillée STABP.

arbre STABP(graphe G, groupe R)

```
1 liste L =trier(E) ; /* liste triée des arêtes en ordre croissant */
2 liste I = ∅ ; /* inhibitions */
3 entier n = |V| ; /* nombre de nœuds à couvrir */
4 ensemble d'arêtes A = ∅ ; /* arêtes agrégées */
5 arbre T_R = MSTC(L,I,n) ; } /* arbre couvrant minimal de G */
6 agrégation(L,T_R,A) ; /* agrégation de nœuds de type R */
7 groupe S = V\R ; /* nœuds de type S */
```

```
8  arbre T_0 = T_R ; /* initialisations */
9  entier dT_0 = d(T_R) ;
10   Pour ∀K ⊆ S tq K ≠ ∅ faire : /* configurations */
11   si (valide(K)) /* si K ne provoque pas de coupe */
12      alors
13         n = |V| − |K| ; /* nombre de nœuds à couvrir */
14         liste I =inhibitions(K) ; /* arêtes interdites */
15         arbre T_R = MSTC(L,I,n) ; } /* arbre couvrant minimal */
16         si d(T_R) < dT_0
17            alors
18               T_0 = T_R ;
19               dT_0 = d(T_R) ;
20         modifier(L,T_R) ; /* propositions BP_3 et BP_4 */
21  fait ;
22  T_0 = T_0 ⋃ {A} ; /* ajout des arêtes agrégées */
23  retourner(T_0) ;
fin.
```

Le calcul sous contraites de l'arbre couvrant minimal est basé sur l'algorithme bien connu de Kruskal. En choisissant les arêtes dans l'ordre croissant des longueurs, l'algorithme doit vérifier si l'une arêtes est sur la liste des inhibitions ou non.

arbre MSTC(liste L, liste I, entier n)

```
1  forêt F = ∅ ; /* forêt des composants connexes construits */
2  entier m = 0 ; /* nombre d'arêtes de F */
3  entête(L) ; /* début de la liste */
4  tant que m < n − 1 faire :
5    arête e = élément(L) ; /* élément courant de L */
6    si (e ∉ I) ; /* l'arête n'est pas inhibée */
7       alors
8          si (F ⋃ {e} n'a pas de cycle)
9             alors
10               F = F ⋃ {e} ;
11               m = m + 1 ;
12         successeur(L) ; /* avancer dans la liste */
13  fait ;
14  arbre A = element(F) ; /* le seul arbre de F */
15  retourner(A) ;
fin.
```

Comme le lemme 3.21 l'indique, le temps de calcul nécessaire de la construction de l'arbre minimal de Steiner avec cette méthode est borné par $O(2^{k-2}(r+k-1)^2)$.

4.2.6 Eclatement de l'arbre de Steiner en forêt

La construction de la forêt joignante minimale à partir de l'arbre minimal de Steiner correspondant est une opération inverse de l'opération de la contraction vue dans la section 4.2.4. Ici, il s'agit de l'éclatement du nœud τ_f de l'arbre de Steiner en $k = |\tau|$ nœuds. Cet éclatement nécessite $O(k)$ manipulations élémentaires.

4.2.7 Création de la forêt déductible maximale d'un membre de la partition

La forêt déductible maximale du problème réduit peut être créée à partir de la forêt joignante minimale à l'aide d'une soustraction. On sait d'après l'algorithme de base FDM (cf. la section 4.1) que pour construire la forêt déductible maximale du composant d'origine (avant les simplifications), il faut ajouter à la forêt les sous-arbres terminaux enlevés par la fonction **réduction** (cf. la section 4.2.3). La soustraction et l'addition peuvent être réalisées par un simple parcours du composant. Cette opération est bornée par $O(|E| + |E|)$.

4.3 Complexité de la construction

Soit T un arbre couvrant du groupe R et τ un ensemble de ses nœuds tels que présentés à la section 3.1. On peut alors résumer les résultats sur la complexité par le théorème suivant.

Théorème 4.1 *Pour construire la forêt déductible maximale de l'arbre T couvrant R relativement à τ, il existe un algorithme dont la complexité est bornée par $O(2^{|\tau|-2}(|R|-1)^2)$.*

Preuve. Utilisons l'algorithme présenté ci-dessus pour la construction de la forêt déductible maximale. Dans le pire des cas, la quasi-partition de l'arbre T donne un composant, appelons le $\tilde{T}_1$, qui contient toutes les nœuds de τ.

Etant donné que les feuilles de T appartiennent à R, $\tilde{T}_1$ contient au plus $|R| - |\tau|$ nœuds internes de type R. Dans ce cas, le plus défavorable pour la construction de l'arbre minimal de Steiner, les autres composants de la quasi-partition ne contiennent chacun qu'un seul nœud de τ et donc ne contiennent pas de forêt déductible.

On ne construit l'arbre minimal de Steiner que pour le composant $\tilde{T}_1$. Pour calculer le pire cas, on suppose que le composant $\tilde{T}_1$ ne permet pas de réduction du problème. La construction de l'arbre minimal de Steiner donne le coût prépondérant de l'algorithme ; les autres fonctions ont une complexité faible, polynomiale. Selon le lemme 3.21, la complexité de la construction de l'arbre minimal de Steiner est ici bornée par $O(2^{|\tau|-2}(|R| - |\tau| + |\tau| - 1)^2)$. ■

5 Conclusions

Plusieurs heuristiques demandent de déterminer l'ensemble des arêtes de longueur maximale d'un arbre de Steiner pouvant être supprimé lors de l'ajout d'un sous-arbre. Dans cet esprit, les travaux de Zelikovsky et de Berman et Ramaiyer portaient sur l'utilisation du

graphe complet du problème initial. Dans ce cas particulier où il s'agit d'un graphe complet, l'ajout d'un sous-arbre à l'arbre couvrant existant implique la suppression d'un ensemble d'arêtes et les arêtes de l'ensemble déductible optimal peuvent être identifiées une par une.

Notre approche généralise le cas de la détermination de la partie déductible maximale d'un arbre couvrant partiel lors de l'ajout d'un autre arbre. Ici, nous montrons que, dans le cas général, la partie déductible est une forêt. Pour déterminer la forêt déductible maximale d'un problème ainsi posé, nous donnons une solution générale qui profite des avantages topologiques du problème d'origine via une technique de séparation et d'évaluation progressive (*divide et impera*). Pour déterminer la forêt déductible maximale, nous proposons également un ensemble de règles de réduction. Le problème réduit puis transformé nous conduit au problème bien connu de Steiner, mais cette fois-ci de taille réduite. Pour résoudre ce dernier problème, nous proposons un algorithme original d'énumération d'arbres couvrants. Bien sûr, étant voisin du problème de Steiner, la complexité de l'algorithme de détermination de la forêt déductible maximale est non polynomiale dans sa généralité. Mais les techniques introduites nous laissent espérer une bonne efficacité des algorithmes proposés.

Références

[1] A.Z. Zelikovsky, *"A 11/6-Approximation Algorithm for the Network Steiner Problem"*, Algorithmica, vol. 9. pp. 463-470, 1993.

[2] Du, Zhang, Feng, *"On Better Heuristic for Euclidean Steiner Minimum Trees"*, Algorithmica, vol. 9. pp. 463-470, 1993.

[3] M. Karpinski, A. Zelikovsky, *"New Approximation Algorithm for the Steiner Tree Problems"*, Journal of Combinatorial Optimizatiom, vol. 13. pp. 47-65, 1997.

[4] P. Berman, V. Ramaiyer, *"Improved Approximations for the Steiner Tree Problem"*, Journal of Algorithm, vol. 17., pp. 381-408, 1994.

[5] M. Molnár, *"Construction d'arbres de diffusion multicast basée sur une modification de l'heuristique de Kruskal"*, RR INRIA no 3587, décembre 1998,
URL= ftp://ftp.inria.fr/INRIA/publication/RR/RR-3587.ps.gz

[6] R. Marie, M. Molnár, *"Arbre couvrant partiel pseudo-optimal pour diffusion multipoint"*, RR INRIA no 3636, mars 1999,
URL= ftp://ftp.inria.fr/INRIA/publication/RR/RR-3636.ps.gz

[7] S.L. Hakimi, *"Steiner's problem in graphs and its implications"*, Network, vol. 1., pp. 113-133, 1971.

[8] E.L. Lawler, *"Combinatorial Optimization: Networks and Matroids"*, Network, Holt, Rinehart and Winston, New York, 1976.

[9] P. Winter, *"Steiner problem in networks: A survey"*, Networks, vol. 17. pp. 129-167, 1987.

[10] A. Balakrishnan, N. R. Patel, *"Problem Reduction Methods and a Tree Generation Algorithm for the Steiner Network Problem"*, Networks, vol. 17. pp. 65-85, 1987.

[11] F. K. Hwang, D. S. Richards, *"Steiner Tree Problems"*, Computer Science Report No. TR-87-21, University of Virginia, 1987.

[12] H.N. Gabow, *"Two Algorithms for Generating Weighted Trees in Order"*,SIAM J. Comput, vol. 6. No 1. pp. 139-150, 1977.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399